Quickstart Guide for Desktop and Mobile Development

What's New in

Adobe AIR 3

O'REILLY®

Adobe Developer Library

Joseph Labrecque

www.it-ebooks.info

www.it-ebooks.info

What's New in Adobe AIR 3

Joseph Labrecque

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

www.it-ebooks.info

What's New in Adobe AIR 3
by Joseph Labrecque

Copyright © 2012 Fractured Vision Media, LLC. All rights reserved.
Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor: Mary Treseler
Production Editor: Dan Fauxsmith
Proofreader: O'Reilly Publishing Services
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Robert Romano

Revision History for the First Edition:

See *http://oreilly.com/catalog/errata.csp?isbn=9781449311087* for release details.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. The image of the Arched or Whistling Duck and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 978-1-449-31108-7

[LSI]

1323097087

www.it-ebooks.info

Adobe Developer Library, a copublishing partnership between O'Reilly Media Inc., and Adobe Systems, Inc., is the authoritative resource for developers using Adobe technologies. These comprehensive resources offer learning solutions to help developers create cutting-edge interactive web applications that can reach virtually anyone on any platform.

With top-quality books and innovative online resources covering the latest tools for rich-Internet application development, the *Adobe Developer Library* delivers expert training straight from the source. Topics include ActionScript, Adobe Flex®, Adobe Flash®, and Adobe Acrobat®.

Get the latest news about books, online resources, and more at *http://adobedeveloperlibrary.com*.

www.it-ebooks.info

www.it-ebooks.info

Table of Contents

Preface ..... **ix**

1. Improvements to the MovieClip and Drawing APIs **1**
Cubic Bezier Curves 1
DisplayObjectContainer.removeChildren() 3
MovieClip.isPlaying 5

2. External Image Capabilities **9**
Enhanced High-Resolution Bitmap Support 9
JPEG-XR Support 11

3. Stage3D: High Performance Visuals **15**
Stage3D Accelerated Graphics Rendering 15
Elements of Stage3D 16
Stage3D Example Using Away3D 18
Stage3D Example Using Starling 20
Tooling Support for Stage3D 24

4. Mobile Advantage: StageText and StageVideo **27**
StageText Native Text Input UI (Mobile) 27
StageVideo Hardware Acceleration (Mobile) 30

5. Video and Audio Enhancements **35**
H.264/AVC Software Encoding 35
Encoding H.264 within AIR 3 36
Reading an H.264 Stream into AIR 3 38
G.711 Audio Compression for Telephony 40

6. Mobile Device Hardware Additions **45**
Camera Position API (Mobile) 45
Device Speaker Control (Mobile) 47

www.it-ebooks.info

Background Audio Playback Support on iOS (Mobile) 49

7. Data Transfer Additions 51
Native JSON (JavaScript Object Notation) Support 51
JSON.parse() 52
JSON.stringify() 54
Socket Progress Events 57

8. Runtime Enhancements 61
ActionScript Native Extensions 61
Captive Runtime Support 67
Android Color Depth Setting (Mobile) 70
Garbage Collection Advice 71

9. Adobe AIR Security 77
Encrypted Local Storage (Mobile) 77
Protected HTTP Dynamic Streaming and Flash Access Content Protection Support for Mobile 80
Secure Random Number Generator 81

Appendix: Additional Resources 85

www.it-ebooks.info

Preface

Introduction to Adobe AIR 3

This book will detail the various enhancements, new functionalities, and general improvements available in this version of the Adobe AIR runtime. Each item is explained in detail, and when possible, a series of screen captures and a full code example will be provided, enabling you to both grasp the new feature in a visual way, and integrate the feature into your own code quickly, based upon example.

AIR, of course, shares many core functionalities with the Adobe Flash Player. During the development cycle between Flash Player 10 and Flash Player 10.1, Adobe rewrote much of the underlying code in order to lay a solid foundation that not only benefited traditional web experiences, but could also be brought over into new areas such as mobile and television. This foundation has served to make both Flash Player 10.1–10.3 and AIR 2.5–2.7 very stable while allowing Adobe to begin adding small features upon each incremental release. In contrast to these incremental versions, with Flash Player 11 and AIR 3 we begin to see the rapid evolution of the Flash Platform runtimes into something not only great at interactive, gaming, media distribution, and enterprise applications...but into something that pushes these areas way beyond their previous limitations.

There is no doubt that mobile application development using the Adobe Flash Platform has become a topic of increased interest in the application developer communities. While there are a number of solutions to cross-compile applications to a variety of mobile platforms using any number of technologies, the ability to do this with such a proven platform is something that most cannot even hope to match. It is very important that AIR evolves in a way which not only showcases why it is so relevant in this new ecosystem, but also why it is (in many cases) the ideal technology platform for advanced interaction on a multitude of devices. With Adobe ramping up the AIR release schedule along with more iterative tooling support in Flash Professional and Flash Builder, not to mention a number of new community partnerships in support of the platform from both independent framework and third-party tooling support, we can expect great things in future incremental releases of AIR 3 and within the entire platform ecosystem.

www.it-ebooks.info

Who This Book Is For

This book is written for both veteran Flash Platform developers curious about enhancements in Adobe AIR 3, as well as those who are entirely new to the Flash Platform. The reader will acquire a solid overview of new features along with usable code examples.

Who This Book Is Not For

This book is not an in-depth study of ActionScript or Adobe AIR internals. Neither is this meant to be an exhaustive overview of complex new features such as *Stage3D* or ActionScript Native Extensions (*ANE*). Entire books will be written which cover such advanced topics. This book will simply provide the reader with a holistic foundation to be built upon using other resources.

Conventions Used in This Book

The following typographical conventions are used in this book:

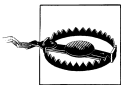
Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

www.it-ebooks.info

This Book's Example Files

You can download the example files for this book from this location:

http://examples.oreilly.com/0636920021681

All code examples are written using pure ActionScript 3, when possible, and are not tied to any framework or IDE. This is to allow the reader to implement the code examples in whichever environment he/she chooses.

The examples are all ActionScript 3 (*AS3*) class files which can be compiled to *AIR, APK, EXE, BAR, IPA*, et cetera, using Flash Professional, Flash Builder, FDT, FlashDevelop, or any other IDE which can be configured to process and output Flash content.

For most of the *mobile* examples with figures, we are setting the `<aspectRatio>` node within the `<initialWindow>` node in the application descriptor file to "landscape" and the `<autoOrients>` node within the `<initialWindow>` node to "false". This is not required, but you may wish to do this yourself when using these examples in order to produce a similar output as is detailed by the figures present in this book.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*What's New in Adobe AIR 3* by Joseph Labrecque (O'Reilly). Copyright 2012 Fractured Vision Media, LLC, 978-1-4493-1108-7."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

How to Use This Book

Development rarely happens in a vacuum. In today's world, email, Twitter, blog posts, co-workers, friends, and colleagues all play a vital role in helping you solve development problems. Consider this book yet another resource at your disposal to help you solve the development problems you will encounter. The content is arranged in such a way

www.it-ebooks.info

that solutions should be easy to find and easy to understand. However, this book does have a big advantage: it is available anytime of the day or night.

Safari® Books Online

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://shop.oreilly.com/product/0636920021681.do

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

www.it-ebooks.info

Acknowledgments

I'd first like to thank my wife, Leslie, and our daughters, Paige and Lily, for being so understanding of the work that I do. It's strange stuff, I know.

Thanks also to Rich Tretola, Chris Griffith, Michelle Yaiser, Brian Rinaldi, Richard Galvan, O'Reilly Media, Adobe Systems, and the Adobe Education Leader and Adobe Community Professional organizations.

www.it-ebooks.info

www.it-ebooks.info

CHAPTER 1

Improvements to the MovieClip and Drawing APIs

Adobe AIR shares much of its core functionality with the Adobe Flash Player runtime. Flash Player began life in the mid-1990s as a web-based media animation and display technology. For much of its history, it has been relied on for graphically intense, functional, and beautiful image rendering and manipulation. With AIR 3, the graphics and vector drawing technology which is so core to Flash Player and inherited by AIR is extended and improved upon in some rather useful ways.

Cubic Bezier Curves

We have an addition to the graphics drawing APIs in this release of AIR which allows simple creation of *Cubic Bezier Curves* without having to do a lot of complex equations on your own, each time you want to draw a new curve. The new `cubicCurveTo()` method takes six arguments to function correctly: a set of `x` and `y` coordinates for the first control point, a similar set for the second control point, and a set of coordinates for the anchor point.

Bezier curves are widely used in computer graphics to model smooth curves through the use of four distinct points: a start point, an end point, and two anchor points which inform the direction and pull of the drawn curve.

The curve will begin wherever the current line is – we can use the `moveTo()` method to precisely position the start point just as is done on other graphics API calls. The two control points influence the curve of the line, and the anchor point will be the end of the drawn curve. This is illustrated visually in Figure 1-1.

www.it-ebooks.info

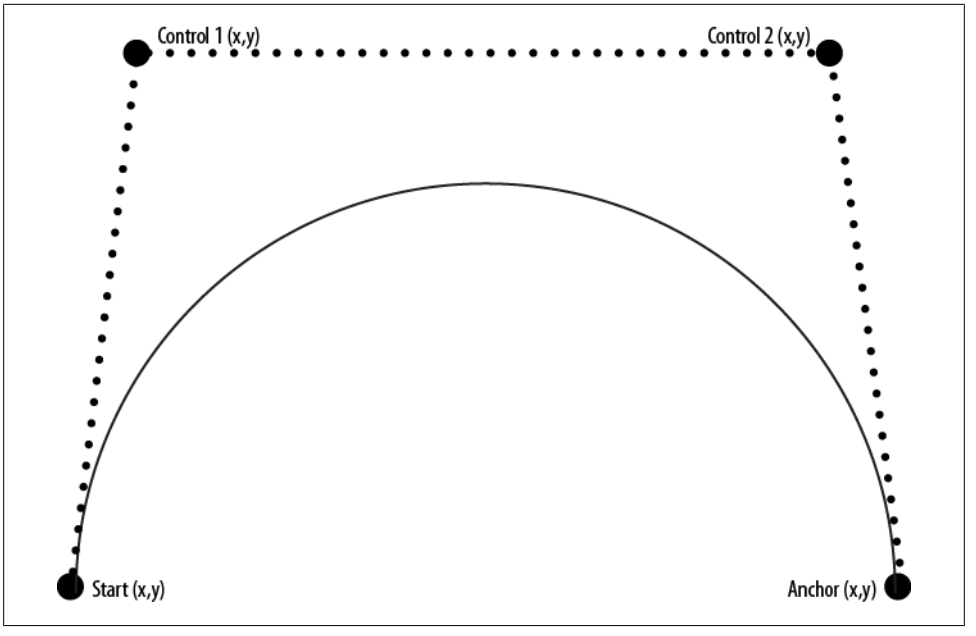


Figure 1-1. How cubic Bezier curves work

In the example below, we create a *Sprite* within which the new `cubicCurveTo()` method is invoked in order to draw a cubic Bezier arc across the stage.

```
package {
    import flash.display.Sprite;

    [SWF(width="600", height="500", backgroundColor="#CCCCCC")]

    public class CubicBezierCurve extends Sprite {

        private var drawingHolder:Sprite;

        public function CubicBezierCurve() {
            generateDisplayObjects();
        }

        protected function generateDisplayObjects():void {
            drawingHolder = new Sprite();
            drawingHolder.graphics.moveTo(20, stage.stageHeight-20);
            drawingHolder.graphics.lineStyle(5,0x000000);

            drawingHolder.graphics.cubicCurveTo(50, 50, stage.stageWidth-50, 50,
stage.stageWidth-20, stage.stageHeight-20);

            addChild(drawingHolder);
        }
```


www.it-ebooks.info

```
        }
    }
```

This will render an application window similar in appearance to Figure 1-2.

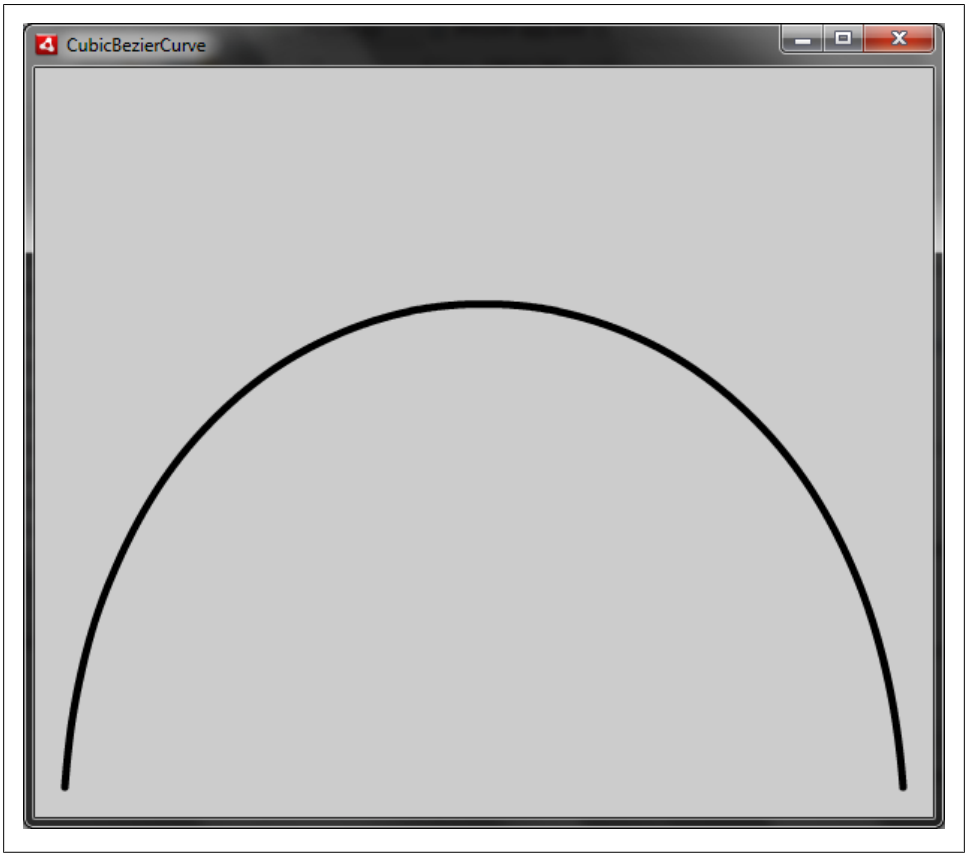


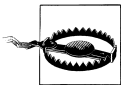
Figure 1-2. Cubic Bezier curve

DisplayObjectContainer.removeChildren()

Previous to AIR 3, if a developer wanted to remove all children from a container object, it was necessary to first determine how many children were present through `DisplayObjectContainer.numChildren` and then loop over each of these child objects, removing them one at a time.

With the `DisplayObjectContainer.removeChildren()` method, one simple command can be used to remove all children of a parent container, making them all available for garbage collection.

www.it-ebooks.info



You'll want to be sure to remove any event listeners or other references to these children before invoking `removeChildren`, else the garbage collector may not be able to totally free the memory allocated to these objects.

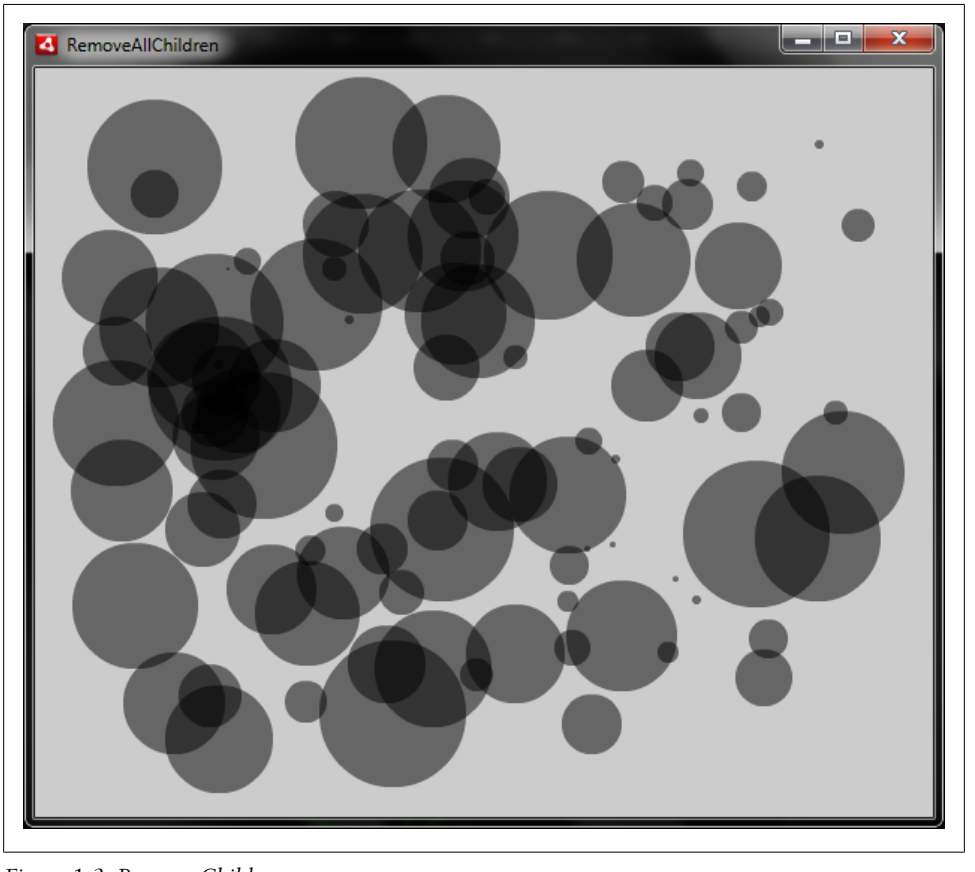


Figure 1-3. Remove Children

In the following example, we will generate a number of dynamic `MovieClip` symbols upon the `Stage`. We add an event listener to the `Stage` as well, listening for a simple `MouseEvent.CLICK` event – which then invokes a method to remove all of these `MovieClips` with one simple command: `stage.removeChildren()`.

```
package {
    import flash.display.MovieClip;
    import flash.display.Sprite;
    import flash.events.MouseEvent;

    [SWF(width="600", height="500", backgroundColor="#CCCCCC")]

    public class RemoveAllChildren extends Sprite {
```


www.it-ebooks.info

```
        public function RemoveAllChildren() {
            generateDisplayObjects();
        }

        protected function generateDisplayObjects():void {
            for(var i:int=100; i>0; i--){
                var childMC:MovieClip = new MovieClip();
              var randX:Number = Math.floor(Math.random() * (1+stage.stageWidth-100))
+ 50;
               var randY:Number = Math.floor(Math.random() * (1+stage.stageHeight-100))
+ 50;
                var randD:Number = Math.floor(Math.random() * 50-10) + 10;
                childMC.x = randX;
                childMC.y = randY;
                childMC.graphics.beginFill(0x000000, 0.5);
                childMC.graphics.drawCircle(0, 0, randD);
                childMC.graphics.endFill();
                this.addChild(childMC);
            }
            stage.addEventListener(MouseEvent.CLICK, removeAllChildren);
        }

        protected function removeAllChildren(e:MouseEvent):void {
            stage.removeChildren();
        }
    }
}
```

MovieClip.isPlaying

It's actually sort of amazing that we haven't had this property in older versions of Flash Player and AIR. `MovieClip` instances are unique in that they contain their own timeline, independent from the main timeline. Often, a developer will want to know whether or not a specific `MovieClip` instance is actually playing or not, and this has traditionally involved monitoring the current frame of the `MovieClip` to determine whether or not it is changing over time.

Making use of this new functionality is very direct, as `MovieClip.isPlaying` is simply a property of every `MovieClip` instance which, when invoked, returns a `Boolean` value of `true` for playing and `false` for stopped. In the following example, we create a `MovieClip` add it to the `DisplayList`, and then write the `isPlaying` property out onto a `TextField`.

```
package {
    import flash.display.MovieClip;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.MouseEvent;
    import flash.text.TextField;
    import flash.text.TextFormat;

    [SWF(width="600", height="500", backgroundColor="#CCCCCC")]
```


www.it-ebooks.info

```
    public class CheckPlaying extends Sprite {

        private var face:MovieClip;
        private var traceField:TextField;

        public function CheckPlaying() {
            generateDisplayObjects();
        }

        protected function generateDisplayObjects():void {
            face = new AngryFace() as MovieClip;
            face.x = stage.stageWidth/2;
            face.y = stage.stageHeight/2;
            face.stop();
            face.addEventListener(MouseEvent.CLICK, toggleFacePlaying);
            addChild(face);

            var defaultFormat:TextFormat = new TextFormat();
            defaultFormat.font = "Arial";
            defaultFormat.size = 26;
            defaultFormat.color = 0xFFFFFF;

            traceField = new TextField();
            traceField.backgroundColor = 0x000000;
            traceField.alpha = 0.7;
            traceField.autoSize = "left";
            traceField.background = true;
            traceField.defaultTextFormat = defaultFormat;
            addChild(traceField);

            stage.addEventListener(Event.ENTER_FRAME, checkPlaying);
        }

        protected function toggleFacePlaying(e:MouseEvent):void {
            if(face.isPlaying){
                face.stop();
            }else{
                face.play();
            }
        }

        protected function checkPlaying(e:Event):void {
            traceField.text = "MovieClip is playing? => " + face.isPlaying;
        }
    }
}
```


www.it-ebooks.info

Figure 1-4. MovieClip.isPlaying

The result of this code can be seen fully rendered in Figure 1-4. When clicking upon the `MovieClip`, its playback is toggled and the `isPlaying` Boolean is measured and written onto the screen.

www.it-ebooks.info

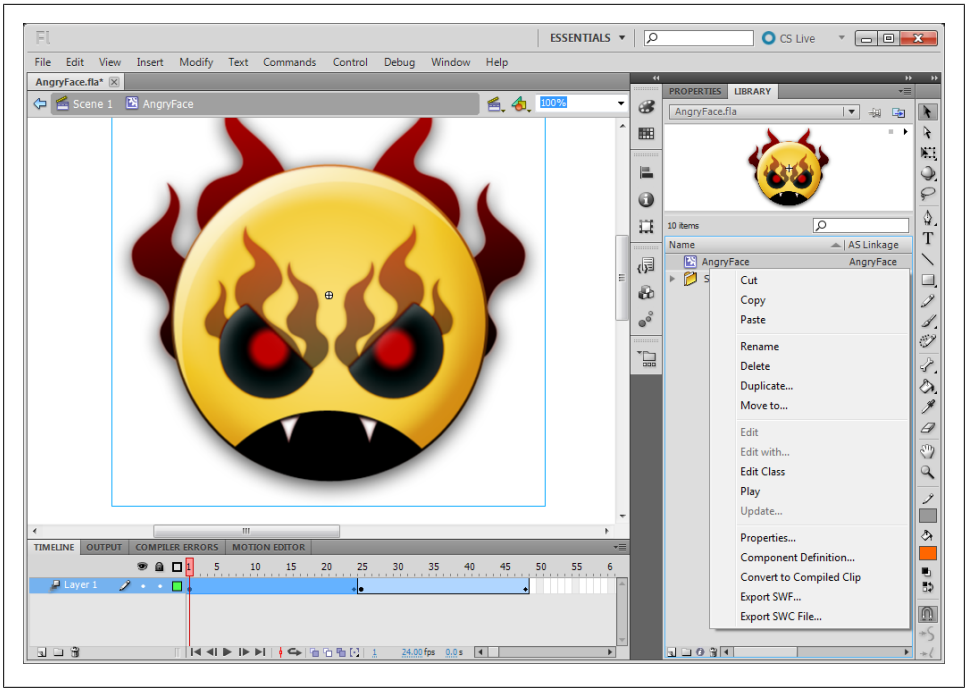


Figure 1-5. Export SWC from Flash Professional

Note that in this example, we are employing a `MovieClip` object that was animated in Flash Professional CS5.5, exported as part of a SWC, and linked into Flash Builder 4.5. There are other ways of doing this, but this method is very direct if you are not working within Flash Professional already.

www.it-ebooks.info

CHAPTER 2

External Image Capabilities

Taking into account the close relationship between AIR and Flash Player, along with Flash Player's focused ability to readily handle vector drawing objects, it is often overlooked how capable the Flash Platform is at utilizing bitmap data through embedded or external image files. Whether using *PNG*, *JPG*, *GIF*, or the new *JPEG-XR* filetype, there is no denying that this imaging technology is extended and improved upon in some rather spectacular ways.

Enhanced High-Resolution Bitmap Support

Loaded `BitmapData` objects have historically been limited to 8,191 total pixels along any side with a total supported resolution of 16,777,215 pixels...which isn't a whole lot when dealing with high resolution images. With the megapixel count of consumer digital cameras breaking well past 10, the need for greater resolution is easily apparent. With AIR 3, these restrictions have been lifted, making this is a feature that can be leveraged through a multitude of project types.

1 megapixel is equal to **1,000,000** pixels.

AIR 2 supports up to **16.777** megapixels.

AIR 3 incudes no such restrictions.

www.it-ebooks.info

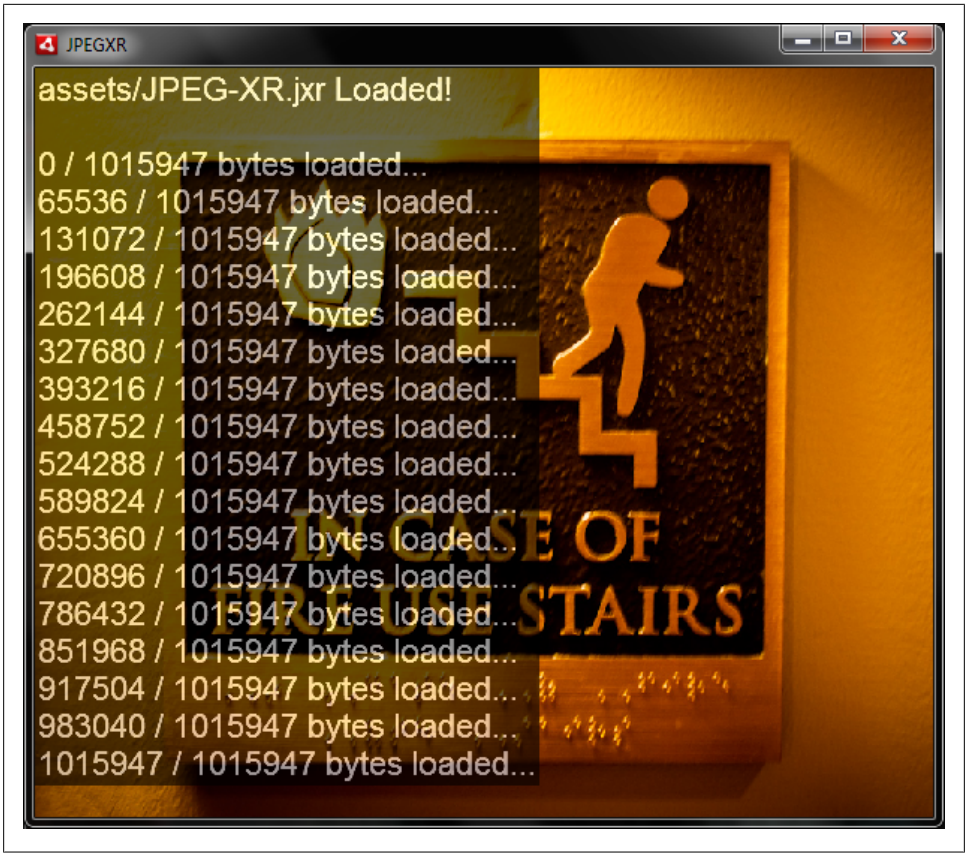

JPEGXR
assets/JPEG-XR.jxr Loaded!
0 / 1015947 bytes loaded...
65536 / 1015947 bytes loaded...
131072 / 1015947 bytes loaded...
196608 / 1015947 bytes loaded...
262144 / 1015947 bytes loaded...
327680 / 1015947 bytes loaded...
393216 / 1015947 bytes loaded...
458752 / 1015947 bytes loaded...
524288 / 1015947 bytes loaded...
589824 / 1015947 bytes loaded...
655360 / 1015947 bytes loaded...
720896 / 1015947 bytes loaded...
786432 / 1015947 bytes loaded...
851968 / 1015947 bytes loaded...
917504 / 1015947 bytes loaded...
983040 / 1015947 bytes loaded...
1015947 / 1015947 bytes loaded...
IN CASE OF
FIRE USE STAIRS

Figure 2-1. High-resolution bitmap

We don't actually need to do anything to enable support for this behavior, as it is built into the AIR runtime itself. In the following example, we'll use the `Loader` class to bring a high-resolution image into an AIR project:

```
package {
    import flash.display.Loader;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.net.URLRequest;
    import flash.text.TextField;
    import flash.text.TextFormat;
    import flash.events.ProgressEvent;

    [SWF(width="600", height="500", backgroundColor="#CCCCCC")]

    public class HighRes extends Sprite {

        private var imageLoader:Loader;
        private var traceField:TextField;
```


www.it-ebooks.info

```
        public function HighRes() {
            generateDisplayObjects();
        }

        protected function generateDisplayObjects():void {
            imageLoader = new Loader();
          imageLoader.contentLoaderInfo.addEventListener(Event.COMPLETE, imageLoaded);
            imageLoader.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS,
imageProgress);
            imageLoader.load(new URLRequest("assets/highres.jpg"));
            addChild(imageLoader);

            var defaultFormat:TextFormat = new TextFormat();
            defaultFormat.font = "Arial";
            defaultFormat.size = 22;
            defaultFormat.color = 0xFFFFFF;

            traceField = new TextField();
            traceField.backgroundColor = 0x000000;
            traceField.alpha = 0.6;
            traceField.autoSize = "left";
            traceField.background = true;
            traceField.defaultTextFormat = defaultFormat;
            addChild(traceField);
        }

        protected function imageProgress(e:ProgressEvent):void {
            traceField.appendText(e.bytesLoaded + " / " + e.bytesTotal + " bytes
loaded...\n");
        }

        protected function imageLoaded(e:Event):void {
            imageLoader.height = stage.stageHeight;
            imageLoader.scaleX = imageLoader.scaleY;

            traceField.text = "Loaded image is " + e.target.width + " x " +
e.target.height + " pixels =>\nThat's " + e.target.width*e.target.height + " total
pixels!\n\n" + traceField.text;
        }

    }
}
```

JPEG-XR Support

AIR 3 includes expanded support for still image file formats. Previous versions of AIR include support for the following image file formats: *GIF*, *JPEG*, and *PNG* – with any other files relying upon external code libraries for interpretation. The recent addition of *JPEG-XR* (International Standard ISO/IEC 29199-2) brings a new image file format to AIR which boasts more efficient compression than *JPG*, along with both lossy and

www.it-ebooks.info

lossless compression options. Like the *PNG* format, *JPEG-XR* also includes a full alpha channel.

You may be wondering how to generate *JPEG-XR* files, since many popular tools (including Adobe Photoshop) do not support the export or conversion to *.JXR* natively. I've found the Windows-only tool **Paint.NET** (*http://paint.net/*) along with the **JPEG XR plugin** (*http://pdnjpegxrplugin.codeplex.com/*) to be most useful in converting images to *JPEG-XR*.

Many conversion programs actually leave out certain bytes which are necessary for the file to load into the runtime, due to security concerns.

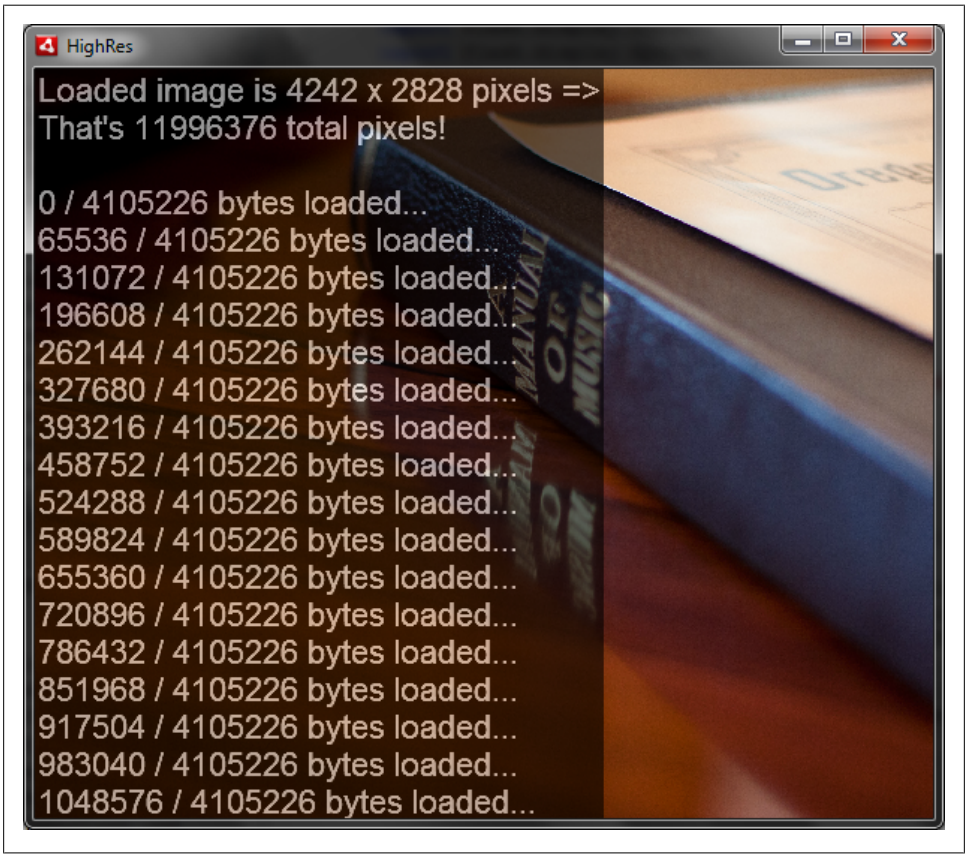


Figure 2-2. JPEG-XR Support

To load a *JPEG-XR* file into AIR, you perform the same set of actions that are necessary for any external image to be loaded into a project:

www.it-ebooks.info

```
package {
    import flash.display.Loader;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.ProgressEvent;
    import flash.net.URLRequest;
    import flash.text.TextField;
    import flash.text.TextFormat;

    [SWF(width="600", height="500", backgroundColor="#CCCCCC")]

    public class JPEGXR extends Sprite {

        private var imageLoader:Loader;
        private var traceField:TextField;
        private const JXR_PATH:String = "assets/JPEG-XR.jxr";

        public function JPEGXR() {
            generateDisplayObjects();
        }

        protected function generateDisplayObjects():void {
            imageLoader = new Loader();
          imageLoader.contentLoaderInfo.addEventListener(Event.COMPLETE, imageLoaded);
            imageLoader.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS,
imageProgress);
            imageLoader.load(new URLRequest(JXR_PATH));
            addChild(imageLoader);

            var defaultFormat:TextFormat = new TextFormat();
            defaultFormat.font = "Arial";
            defaultFormat.size = 22;
            defaultFormat.color = 0xFFFFFF;

            traceField = new TextField();
            traceField.backgroundColor = 0x000000;
            traceField.alpha = 0.6;
            traceField.autoSize = "left";
            traceField.background = true;
            traceField.defaultTextFormat = defaultFormat;
            addChild(traceField);
        }

        protected function imageProgress(e:ProgressEvent):void {
            traceField.appendText(e.bytesLoaded + " / " + e.bytesTotal + " bytes
loaded...\n");
        }

        protected function imageLoaded(e:Event):void {
            imageLoader.height = stage.stageHeight;
            imageLoader.scaleX = imageLoader.scaleY;

            traceField.text = JXR_PATH + " Loaded!\n\n" + traceField.text;
        }
```


www.it-ebooks.info

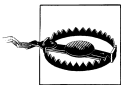
```
        }
    }
```


www.it-ebooks.info

CHAPTER 3

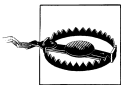
Stage3D: High Performance Visuals

The single most written about feature of AIR 3 would definitely be the new accelerated graphics rendering engine available through *Stage3D* (previously known by the code-name "*Molehill*"). This advanced rendering architecture can be used in rendering both 2D and 3D visual objects within AIR through direct use of these APIs or by implementation of one of the many engines and frameworks that have been built on top of these APIs.



To use `Stage3D` in AIR, we must set `<renderMode>direct</renderMode>` within the application descriptor file.

The main benefit of using the `Stage3D` APIs is that everything rendered using `Stage3D` on supported system configurations will be rendered directly through the system *GPU* (Graphics Processing Unit). This allows the GPU to assume total responsibility for these complex visual rendering tasks, while the *CPU* (Central Processing Unit) remains available for other functions.



In the case that rendering `Stage3D` using hardware is not available on a particular system, the `Stage3D` view will be rendered using software as a fallback.

Stage3D Accelerated Graphics Rendering

The new `flash.display.Stage3D` class works very similar to `flash.media.StageVideo` in how it behaves as a display object within AIR. Just like `StageVideo`, `Stage3D` is never added to the AIR `DisplayList` but rather exists separately from that stack of objects. As in the case of `StageVideo` usage, the `DisplayList` appears above `Stage3D` in the visual stacking order.

www.it-ebooks.info

It's important to note that `Stage3D` does not in any way deprecate or interfere with the "2.5D" capabilities introduced in AIR 2. Those APIs are used with objects added to the traditional `DisplayList`, while the new `Stage3D` APIs are entirely separated from that.

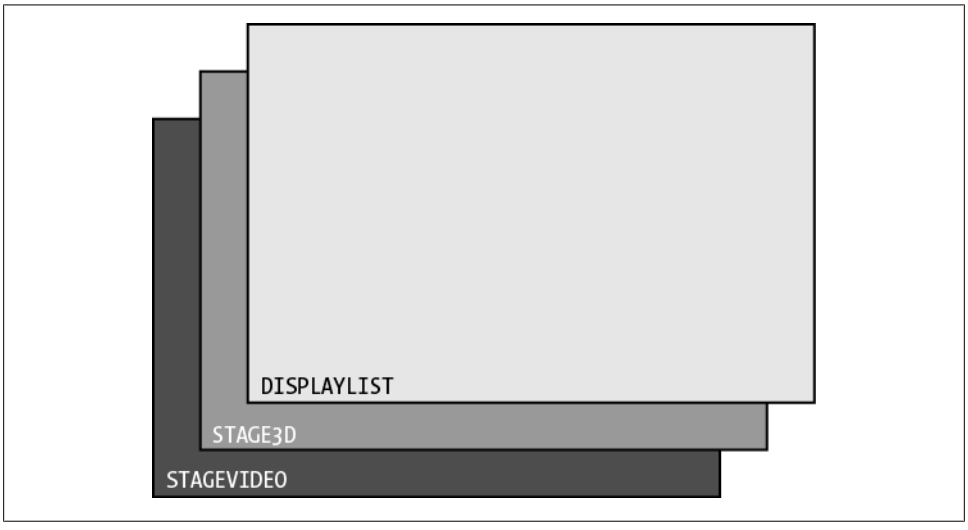
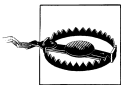


Figure 3-1. Stage3D sits between StageVideo and the traditional DisplayList

This will, no doubt remain one of the most deep and complex sets of classes that an ActionScript developer will come across for some years to come. Thankfully, Adobe has made the wise decision of providing early access of these new APIs to both rendering engine and tooling product creators.



`Stage3D` is currently supported on the desktop only. Mobile `Stage3D` will be supported in a future AIR release.

Elements of Stage3D

As mentioned above, `Stage3D` itself is rather low level in its implantation and quite difficult to work with for most ActionScript developers because of this. If you haven't worked in a 3D programming environment before, many of the terms and objects that are necessary to get this working will seem quite foreign in relation to your normal workflow.

www.it-ebooks.info

For an example of how to leverage these raw APIs, I suggest that the reader visit Thibault Imbert's website at *http://www.bytearray.org/* for a number of `Stage3D` examples and a much deeper information pool than we will get into here.

To get a simple example of `Stage3D` set up and rendering within an AIR application, there are a number of core classes to import, as can be seen below:

```
import flash.display.Stage3D;
import flash.display3D.Context3D;
import flash.display3D.Context3DProgramType;
import flash.display3D.Context3DTriangleFace;
import flash.display3D.Context3DVertexBufferFormat;
import flash.display3D.IndexBuffer3D;
import flash.display3D.Program3D;
import flash.display3D.VertexBuffer3D;
import flash.geom.Matrix3D;
```

When working in `Stage3D`, we have to work with *vertex* and *fragment shaders* in order to render anything upon the `Stage3D` view. For those unfamiliar with the term, *shaders* are low-level software instructions that are used to calculate rendering effects on the system GPU. In fact, these instructions are used to directly program the graphics rendering pipeline or the GPU. *Vertex shaders* affect the direct appearance of a visual element, while *fragment shaders* manage element surface details.

Adobe Pixel Bender 3D allows the production of *vertex* and *fragment shaders* that run on 3D hardware to generate output images. These kernels operate on 3D objects and affect their appearance.

To actually create and render any *shaders*, you'll also need to use a new language called *AGAL* (Adobe Graphics Assembly Language). AGAL is very, very low level and not for the faint of heart. Traditional ActionScript developers will most likely struggle with AGAL, but those familiar with working in other environments such as *OpenGL* or any general *Assembly* language should feel right at home. In either case, the recommended approach to working with `Stage3D` is to use one of the many higher-level frameworks that are available.

While `Stage3D` has a large number of 3D frameworks which utilize it in the creation and rendering of complex 3D graphics within Flash Player, the rendering surface can actually be used for any 3D or even 2D content which utilizes it in enabling an accelerated visual experience.

The basic setup to getting `Stage3D` working in an ActionScript project is to perform the following actions:

www.it-ebooks.info

- Request a `Context3D` object through the `stage.stage3Ds` array.
- Once the `Context3D` object is ready, we can then set up `Context3D` to whatever specifications we have, including our `IndexBuffer3D` and `VertexBuffer3D` objects.
- We then use AGAL to create our various *shaders* to use within a `Program3D` object.
- Finally, all of this is processed through a render loop (`Event.ENTER_FRAME`) and rendered to the `Stage3D` object via `Context3D` and a set of `Program3D` and `Matrix3D` object controls.

If this sounds complicated, that's because it is! The process outlined above and the array of complexities associated with it are really meant for those who wish to build their own frameworks and engines upon a `Stage3D` foundation. In the next section, we'll have a look at how to actually use one of these 3D frameworks to render some content within Flash Player.

There is a project hosted on **Google Code** called **EasyAGAL** which aims to simplify the creation of AGAL for `Stage3D`. The project can be acquired from *http://code.google.com/p/easy-agal/*

Stage3D Example Using Away3D

Thankfully, we don't need to deal with direct APIs and AGAL unless we actually want to. There are a number of very robust, complete 3D frameworks that can be used as high-level alternatives to the AIR `Stage3D` APIs. In this example, we will have a look at a simple implementation using `Away3D` to render an animated primitive using `Stage3D`.

I would encourage those who are curious to perform a basic rendering like this using the direct APIs first, and then compare that with the Away3D implementation. The differences will be quite apparent in how simple a framework like Away3D distills the APIs into a highly usable form.

Before running the example below, you will want to be sure to download the proper Away3D framework code from *http://away3d.com/* for use within your project.

As can be seen in the code below, all we need to do for this to work is to create an instance of `View3D`, generate objects such as the `WireframeCube` primitive, and add these objects to the `View3D.scene` property. Now all we must do is render the `View3D`. This is normally done by creating what is known as a render loop using `Event.ENTER_FRAME` and then executing `View3D.render()` within a method invoked by that event. Upon every iteration of the render loop, we have the opportunity to adjust our object properties. In our example, we adjust the `rotationX` and `rotationY` properties of our `WireframeCube` primitive to create 3D animation.

www.it-ebooks.info

```
package {
    import away3d.containers.View3D;
    import away3d.primitives.WireframeCube;

    import flash.display.Sprite;
    import flash.events.Event;

    [SWF(width="600", height="500", backgroundColor="#CCCCCC")]

    public class SimpleAway3D extends Sprite {

        private var view3D:View3D;
        private var wireframeCube:WireframeCube;

        public function SimpleAway3D() {
            generate3D();
        }

        private function generate3D():void {
            var size:Number = 250;
            wireframeCube = new WireframeCube(size, size, size, 0x24ff00, 5);

            view3D = new View3D();
            view3D.scene.addChild(wireframeCube);

            addChild(view3D);
            addEventListener(Event.ENTER_FRAME, renderLoop);
        }

        protected function renderLoop(e:Event):void {
            wireframeCube.rotationX += 1;
            wireframeCube.rotationY += 3;
            view3D.render();
        }

    }
}
```

Running the above code will produce a wireframe cube slowly rotating along the `x` and `y` axis. `Away3D` comes packaged with a lot of different primitives and materials that can be used in rendering 3D content. This example just scratches the surface of what one might do with such an extensive framework.

www.it-ebooks.info

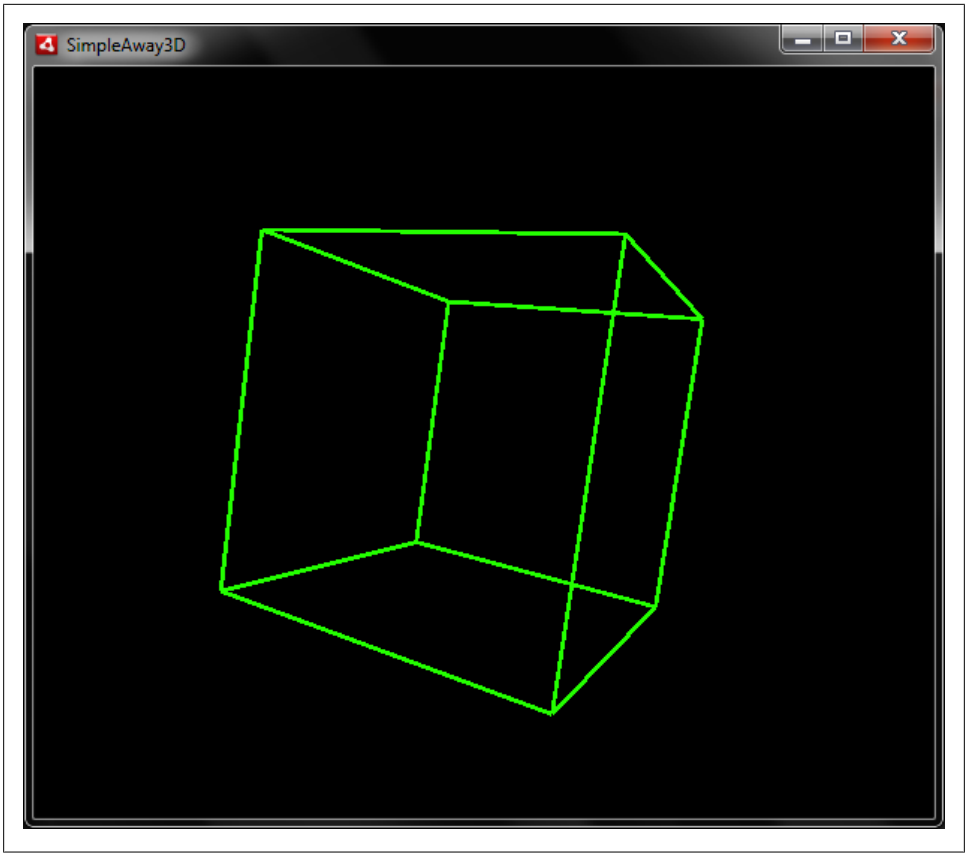


Figure 3-2. WireFrameCube primitive rendered using Away3D

`Away3D` is just one of many ActionScript frameworks which utilize `Stage3D`. These frameworks are meant to provide high-level access to powerful display technology and each has its strengths and weaknesses. Experiment with a number of these frameworks to discover what will work best in your particular project.

A list of `Stage3D` frameworks and libraries is included in Appendix of this book.

Stage3D Example Using Starling

Starling (*http://starling-framework.org/*) is a open source effort begun by **Adobe** and the **Sparrow Framework** (*http://www.sparrow-framework.org/*) to create a 2D framework for `Stage3D` which emulates the traditional `DisplayList` that Flash Platform de-

www.it-ebooks.info

velopers are so used to. In fact, developers can use concepts that they are familiar with, such as `Sprite`, `MovieClip`, and `TextField` in a very similar way to how these objects would be used with native Flash and AIR classes.

Starling is a direct port of the Sparrow framework for iOS, which mimics the Flash `DisplayList` APIs.

The Starling framework can be freely acquired from *http://github.com/PrimaryFeather/Starling---Framework/* and weighs in at only 80k – very lightweight. Since it is an open source project, the community can contribute and help grow the framework.

In this quick example, we will create a simple *Quad* and cause it to continuously rotate clockwise. First, we must set up our Starling classes through the main application class. The important thing here is that we create a new instance of `starling.core.Starling` and pass in a class called `Game` which will contain the remainder of our code. We also pass in a reference to the current `Stage`. The final step is to invoke `Starling.start()` to get things going.

```
package {
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;

    import starling.core.Starling;

    [SWF(width="600", height="500", backgroundColor="#000000")]

    public class SimpleStarling extends Sprite {

        private var starlingBase:Starling;

        public function SimpleStarling() {
            super();
            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            performOperations();
        }

        protected function performOperations():void {
            starlingBase = new Starling(Game, this.stage);
            starlingBase.antiAliasing = 2;
            starlingBase.start();
        }

    }

}
```


www.it-ebooks.info

Now that we have set up Starling, we have to create the `Game` class which it uses upon initialization. All of our rendering will live inside of this `Game.as` class included within the same package as our main application class in this example.

Initially, we want to be sure that our class is added to the `Stage` and ready to perform display functions for us. To do this, we add an event listener of type `Event.ADDED_TO_STAGE`. Once this event fires, we are safe to begin drawing out our visual objects using Starling classes.

Note that even though we are using familiar classes like `Sprite` and `Event`, we are using the Starling versions of these classes - not the core Flash classes.

Here, we now set up our *Quad*. A quad is basically two triangles which link together to form a square plane. We will set this up in such a way that its position is at the center of the `Stage` with a transform point (pivot) at its center. This will allow us to rotate around the center point instead of the upper left, which is default. Using `Quad.setVertexColor()`, we set different shades of green as gradient points.

Finally, we set up the render loop which is invoked through `Event.ENTER_FRAME`. This is where any change over time should occur, and in this case, it does a simple clockwise rotation of the *Quad*.

```
package {
    import starling.display.Sprite;
    import starling.display.Quad;
    import starling.events.Event;

    public class Game extends Sprite {

        private var quad:Quad;

        public function Game() {
            this.addEventListener(Event.ADDED_TO_STAGE, onStageReady);
        }

        protected function onStageReady(e:Event):void {
            quad = new Quad(300, 300);
            quad.pivotX = 150;
            quad.pivotY = 150;
            quad.setVertexColor(0, 0x00ff18);
            quad.setVertexColor(1, 0x2dcb3b);
            quad.setVertexColor(2, 0x00ff18);
            quad.setVertexColor(3, 0x2dcb3b);
            quad.x = (stage.stageWidth/2);
            quad.y = (stage.stageHeight/2);
            this.addChild(quad);
            this.addEventListener(Event.ENTER_FRAME, renderLoop);
        }
```


www.it-ebooks.info

```
            protected function renderLoop(e:Event):void {
                quad.rotation += 0.02;
            }

        }

    }
```

When we compile and run this code on the desktop, we can see how simple using accelerated 2D graphics with `Stage3D` can be thanks to this fabulous framework.

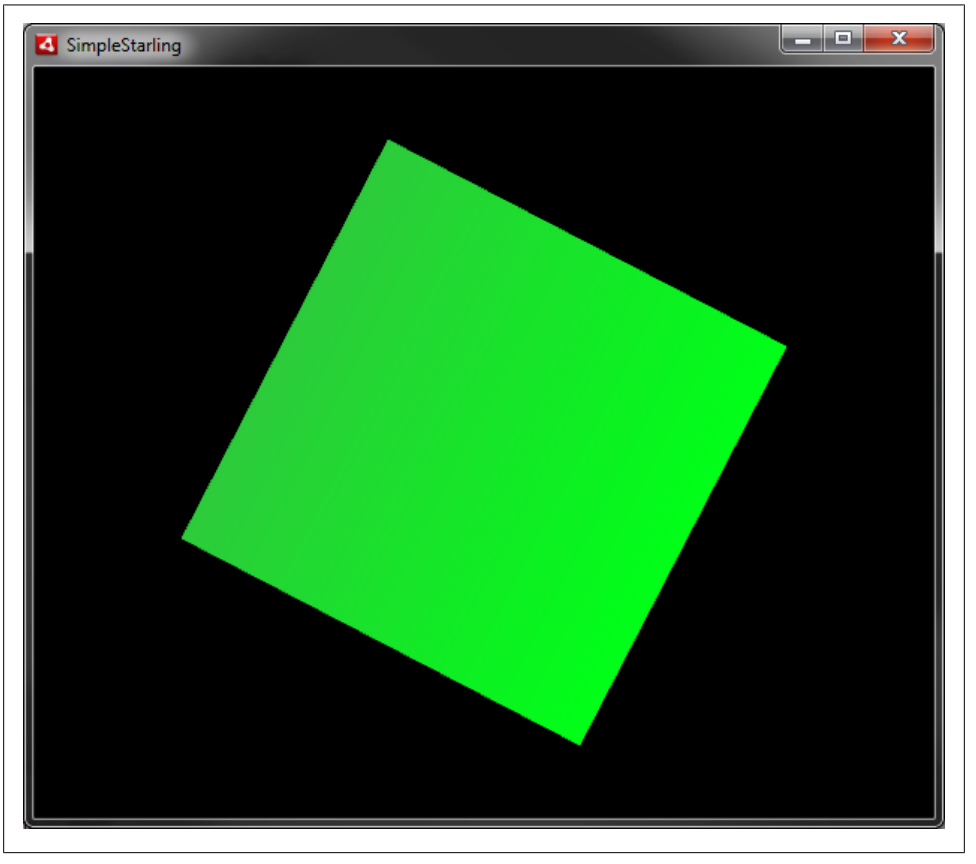


Figure 3-3. Simple Quad render and rotation using Starling

Read all about the Starling framework in Thibault Imbert's book *Introducing Starling* [http://byearray.org/] – just like Starling itself, this book is free!

www.it-ebooks.info

Tooling Support for Stage3D

Not only does `Stage3D` have the support of many 3D frameworks, but a variety of tooling products have also embraced this new functionality. Most notable of these, are the **Unity** development environment and **Flare3D Studio**.

Unity

Unity has built in support for `Stage3D`, going so far as to export directly to a compiled *SWF* which can be nearly identical to an export to the Unity, depending upon supported features. These features include *physics*, *lightmapping*, *occlusion culling*, *custom shaders*, *lightprobes*, *particle systems*, *navigation meshes*, and more! This is truly an incredible development where Flash and AIR gaming is concerned as Unity is such a great gaming engine and editor environment, already in use by many game developers targeting a variety of diverse platforms.

After rendering Unity content for Flash Player, developers should be able to build upon that content within larger Flash Player and AIR projects. One use for this would be to create a robust menuing system for a game.

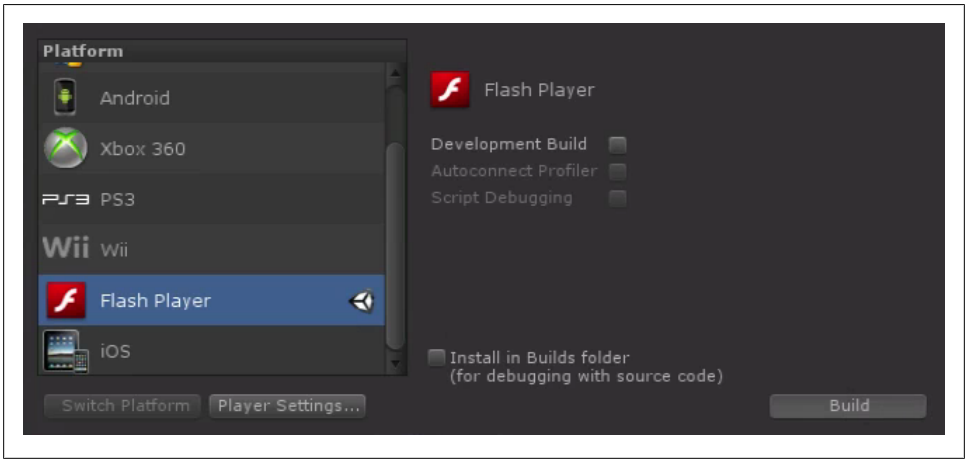


Figure 3-4. Unity3D build settings

Flare3D Studio

Also of note is Flare3D Studio – a 3D design environment build using Flash Platform tooling and distributed using the AIR runtime! It is excellent to see such excitement and collaboration in the industry around `Stage3D` from all of these different players.

www.it-ebooks.info

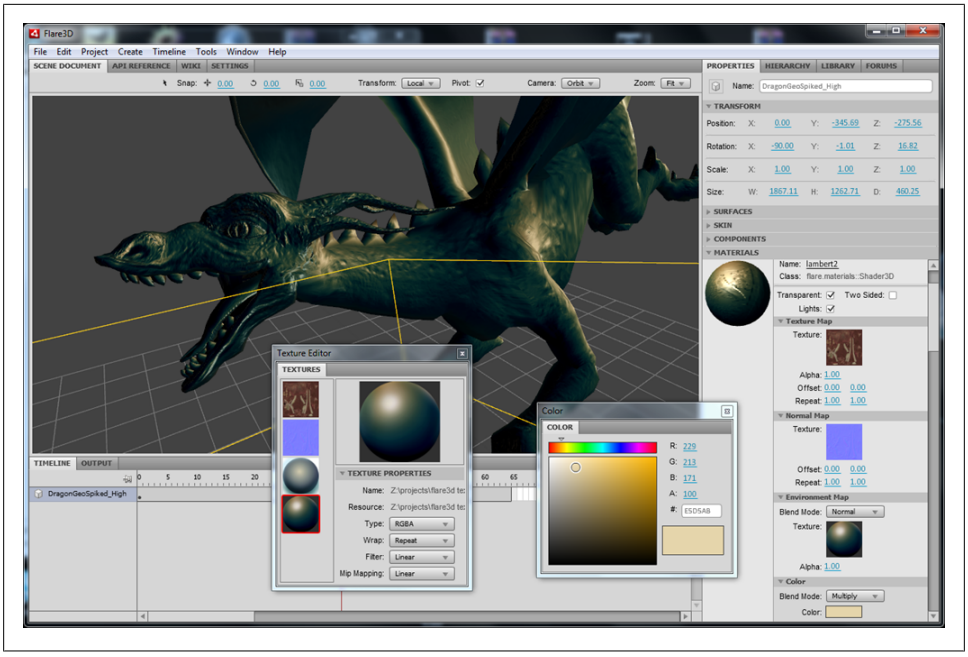


Figure 3-5. Flare3D Studio

I gather that we have much to look forward to in terms of improved tooling from Adobe, Unity, Flare3D, and perhaps other parties as well.

www.it-ebooks.info

www.it-ebooks.info

CHAPTER 4

Mobile Advantage: StageText and StageVideo

With AIR becoming a very real choice when developing for mobile operating systems like iOS and Android, it certainly helps to have the platform features necessary to take advantage of underlying system hardware acceleration and other constructs such as the rich text input frameworks available on a device. AIR 3 extends the integration of the runtime into some of these underlying system functionalities.

StageText Native Text Input UI (Mobile)

The new `flash.text.StageText` class in AIR 3 allows ActionScript developers to tap directly into the text input mechanisms of the underlying operating system, exposing functionality such as auto-correct, auto-capitalize, and the specific type of virtual keyboard that is presented to the user.

Similar to `StageWebView` and `StageVideo`, `StageText` is not part of the traditional `DisplayList` and must be dealt with in its own way. Native text input fields created with `StageText` are always rendered above the `DisplayList` and will appear over any other content being rendered on the `Stage`.

Note that `StageText` does not include a background or border decoration. This must be supplied through the `DisplayList` as demonstrated below.

To create a `StageText` instance, we will first draw out any background elements necessary and add them to the `DisplayList`. In this case, we draw out a simple box using the graphics API. This will provide the user with an indicator which they can tap to provide focus to the overlaying `StageText` object. We then define a new `StageTextInitOptions` instance and set our multiline preference to “true” or “false”.

www.it-ebooks.info

Once that has all been set up, we instantiate a new `StageText` object and begin to define the properties of this object. Properties can include such things as whether to auto-capitalize the text, make auto-correct available for the user, set the return type label, or even the specific variant of soft keyboard used for this particular field.

`flash.text.SoftKeyboardType` options include:

```
SoftKeyboardType.CONTACT
SoftKeyboardType.DEFAULT
SoftKeyboardType.EMAIL
SoftKeyboardType.NUMBER
SoftKeyboardType.PUNCTUATION
SoftKeyboardType.URL
```

Finally we assign the current `Stage` to the `StageText.stage` property and a `Rectangle` is assigned to the `StageText.viewport` property to determine positioning and size.

```
package {
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.geom.Rectangle;
    import flash.text.ReturnKeyLabel;
    import flash.text.SoftKeyboardType;
    import flash.text.StageText;
    import flash.text.StageTextInitOptions;

    [SWF(backgroundColor="#CCCCCC")]

    public class MobileStageText extends Sprite {

        private var decoration:Sprite;
        private var stageText:StageText;
        private var stageTextInitOptions:StageTextInitOptions;

        public function MobileStageText() {
            super();
            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            generateDisplayObjects();
        }

        protected function generateDisplayObjects():void {
            decoration = new Sprite();
            decoration.graphics.lineStyle(4, 0x000000, 1);
            decoration.graphics.beginFill(0xFFFFFF, 1);
            decoration.graphics.drawRect(6, 44, stage.stageWidth-12, 70);
            this.addChild(decoration);

            stageTextInitOptions = new StageTextInitOptions(false);
            stageText = new StageText(stageTextInitOptions);
```


www.it-ebooks.info

```
            stageText.softKeyboardType = SoftKeyboardType.DEFAULT;
            stageText.returnKeyLabel = ReturnKeyLabel.DONE;
            stageText.autoCorrect = true;
            stageText.fontSize = 40;
            stageText.color = 0x440000;
            stageText.fontWeight = "bold";
            stageText.stage = this.stage;
            stageText.viewPort = new Rectangle(10, 50, stage.stageWidth-20, 70);
        }

    }
}
```

Note that with this example, unlike the other mobile AIR examples in this book, we will set the orientation to "portrait" within our application descriptor file in order to provide the soft keyboard with enough room to exist on the screen along with the `StageText` instance itself.

```
<aspectRatio>portrait</aspectRatio>
```

When this code is compiled to a mobile device, we can see the result of a native text interaction as seen in Figure 4-1.

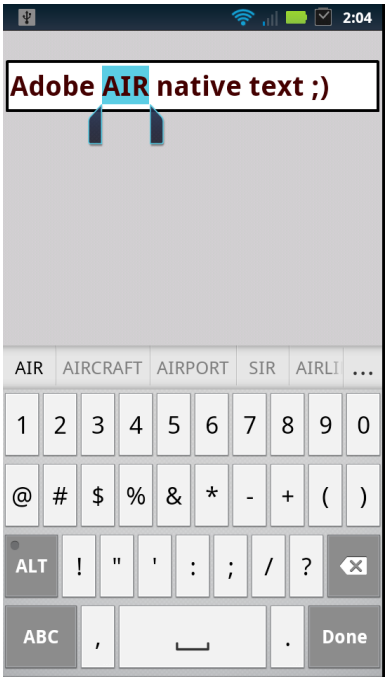


Figure 4-1. Native text on Android

www.it-ebooks.info

To get around some of the complexities associated with using `StageText`, Christian Cantrell has developed the `NativeText` wrapper which assists with a lot of the difficulties one can encounter when using these APIs. `NativeText` can be downloaded from *https://github.com/cantrell/StageTextExample*.

StageVideo Hardware Acceleration (Mobile)

`StageVideo` is one of those things that's been available with Flash Player (along with AIR for TV) for some time now - but is just beginning to move into other areas of the Flash Platform. Using the `StageVideo` class on supported devices allows a developer to leverage the hardware acceleration capabilities of the device for increases in performance, decreased CPU usage, and more efficient battery life when displaying video media using AIR 3.

`StageVideo` is supported on Android 3.1, BlackBerry Tablet OS, and iOS mobile operating systems only.

The `flash.media.StageVideo` class works very similar to `flash.media.Video` in its purpose and function. In fact, `StageVideo` supports the exact same codecs and formats as the `Video` object.

It's important to note that `StageVideo` does not in any way deprecate or interfere with the traditional `Video` object. This can be thought of as simply a hardware accelerated version of that object which sits below the `DisplayList`.

www.it-ebooks.info

Figure 4-2. StageVideo sits directly below the traditional DisplayList

To use `StageVideo` within a mobile AIR project, we must import the `flash.media.StageVideo` class. Note that we do not actually create an instance of this class; instead, if we discover that `StageVideo` is available on a particular device through listening to the `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY` event on the `Stage`, we can the move ahead.

We next assign a `StageVideo` object to a position in the `stageVideos` property of the `Stage` through array syntax (`stage.stageVideos[0]`). To actually play video using the `StageVideo` class, we do this using the exact same methods we normally would when using the standard `Video` class. This is excellent, as it provides an easy fallback in case the `StageVideo` is not available.

Note that the length of the `stage.stageVideos` array will vary by device, depending upon capability, though if `StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY` reports that `StageVideo` is available, we can be sure that we have at least one position available for video playback.

```
package {
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.events.NetStatusEvent;
    import flash.events.StageVideoAvailabilityEvent;
    import flash.media.StageVideo;
    import flash.net.NetConnection;
    import flash.net.NetStream;
    import flash.text.TextField;
    import flash.text.TextFormat;
```


www.it-ebooks.info

```
import flash.geom.Rectangle;

[SWF(backgroundColor="#000000")]

public class StageVideoMobile extends Sprite {

    private var traceField:TextField;
    private var stageVideo:StageVideo;
    private var connection:NetConnection;
    private var stream:NetStream;
    private var streamClient:Object;

    public function StageVideoMobile() {
        super();
        stage.scaleMode = StageScaleMode.NO_SCALE;
        stage.align = StageAlign.TOP_LEFT;

        generateDisplayObjects();
        performOperations();
    }

    protected function generateDisplayObjects():void {
        var defaultFormat:TextFormat = new TextFormat();
        defaultFormat.font = "Arial";
        defaultFormat.size = 30;
        defaultFormat.color = 0xFFFFFF;

        traceField = new TextField();
        traceField.backgroundColor = 0x000000;
        traceField.alpha = 0.7;
        traceField.width = stage.stageWidth;
        traceField.background = true;
        traceField.defaultTextFormat = defaultFormat;
        addChild(traceField);
    }

    protected function performOperations():void {

stage.addEventListener(StageVideoAvailabilityEvent.STAGE_VIDEO_AVAILABILITY,
availabilityChanged);
    }

    protected function availabilityChanged(e:StageVideoAvailabilityEvent):void {
        if(e.availability == "available" && stageVideo == null){
            stageVideo = stage.stageVideos[0];
            createConnection();
        }
        traceField.text = "StageVideo => " + e.availability + "\n";
    }

    protected function createConnection():void {
        streamClient = new Object();
        streamClient.onBWDone = onBWDone;
        streamClient.onMetaData = onMetaData;
        streamClient.onXMPData = onXMPData;
```


www.it-ebooks.info

```
            streamClient.onPlayStatus = onPlayStatus;

            connection = new NetConnection();
            connection.client = streamClient;
            connection.addEventListener(NetStatusEvent.NET_STATUS, monitorStatus);
            connection.connect(null);

            beginStreaming();
        }

        protected function monitorStatus(e:NetStatusEvent):void {
            if(e.info.code == "NetStream.Buffer.Full"){
                stageVideo.viewPort = new Rectangle(0, 0, stage.stageWidth,
stage.stageHeight);
            }
        }

        protected function beginStreaming():void {
            stream = new NetStream(connection);
            stream.client = streamClient;
            stream.addEventListener(NetStatusEvent.NET_STATUS, monitorStatus);
            stageVideo.attachNetStream(stream);
            stream.play("assets/FlashRuntimes.mp4");
        }

        public function onBWDone():void {}
        public function onMetaData(e:*):void {}
        public function onXMPData(e:*):void {}
        public function onPlayStatus(e:*):void {}

    }
}
```

So long as this class is compiled for a mobile operating system which has support for `StageVideo`, the video will play back with full hardware acceleration as seen in Figure 4-3.

www.it-ebooks.info

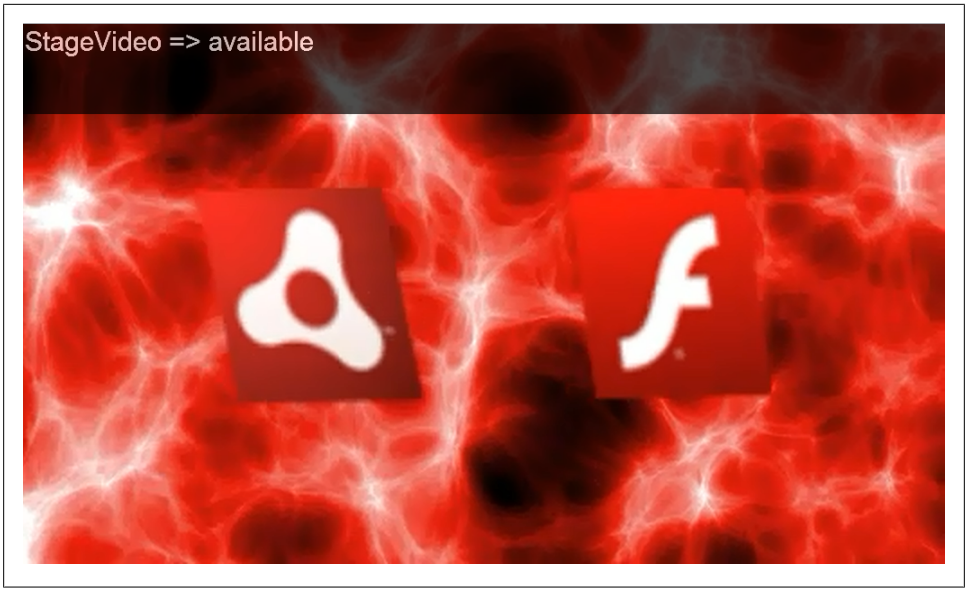


Figure 4-3. StageVideo playback on Tablet OS

www.it-ebooks.info

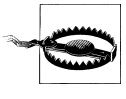
CHAPTER 5

Video and Audio Enhancements

Adobe AIR has always supported the decoding and playback of industry standard `H.264` alongside other codecs such as `VP6` and `Spark`. With AIR 3, developers now have the ability to *encode* captured video to `H.264` in the runtime itself, opening up a lot more possibilities for video capture and distribution applications for the desktop. Along with `H.264` encoding, we also get another audio codec to work with in the form of `G.711`.

H.264/AVC Software Encoding

With AIR 3, developers now have the ability to encode `H.264` video streams within the runtime itself. Prior versions of AIR were able to decode `H.264` and with the additional encoding functionality, a whole other set of applications can be built to record and broadcast in this industry standard format.



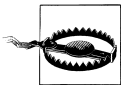
`H.264/AVC` software encoding is only available on the desktop. Mobile devices cannot utilize this feature due to the amount of CPU it takes to encode the streams.

To compile and run the examples included below effectively, it is recommended that you install Flash Media Server.

Note that if you do not have access to a commercial license for Flash Media Server, Adobe does offer a free developer edition with which you can test this and other examples. When doing any serious work through FMS, it is encouraged that you begin with a local development instance such as this.

Flash Media Server developer edition can be acquired from Adobe via *http://www.adobe.com/products/flashmediaserver/*

www.it-ebooks.info



Flash Media Server is available for either Windows or Linux.

Encoding H.264 within AIR 3

To encode a video stream using `H.264` within AIR 3, we must employ the new `H264VideoStreamSettings` class and associated objects. When constructing a `H264VideoStreamSettings` instance for use in our project, we can set the particular profile and level of the encoding. This is useful for targeting certain specific devices which may only support something like baseline encoding. Once we have configured our `H264VideoStreamSettings` instance, we assign it to the `videoStreamSettings` property of a `NetStream` instance and then process the stream as normal.

```
package {
    import flash.display.Sprite;
    import flash.events.NetStatusEvent;
    import flash.media.Camera;
    import flash.media.H264Level;
    import flash.media.H264Profile;
    import flash.media.H264VideoStreamSettings;
    import flash.media.Video;
    import flash.net.NetConnection;
    import flash.net.NetStream;
    import flash.text.TextField;
    import flash.text.TextFormat;

    [SWF(width="600", height="500", backgroundColor="#CCCCCC")]

    public class AVCEncode extends Sprite {

        private var traceField:TextField;
        private var video:Video;
        private var camera:Camera;
        private var connection:NetConnection;
        private var stream:NetStream;
        private var streamClient:Object;

        public function AVCEncode() {
            generateDisplayObjects();
            performOperations();
        }

        protected function generateDisplayObjects():void {
            video = new Video(stage.stageWidth, stage.stageHeight);
            addChild(video);

            var defaultFormat:TextFormat = new TextFormat();
            defaultFormat.font = "Arial";
            defaultFormat.size = 24;
            defaultFormat.color = 0xFFFFFF;

            traceField = new TextField();
```


www.it-ebooks.info

```
            traceField.backgroundColor = 0x000000;
            traceField.alpha = 0.7;
            traceField.autoSize = "left";
            traceField.background = true;
            traceField.defaultTextFormat = defaultFormat;
            addChild(traceField);
        }

        protected function performOperations():void {
            camera = Camera.getCamera();
            camera.setMode(stage.stageWidth, stage.stageHeight, 30);
            camera.setQuality(60000, 80);
            video.attachCamera(camera);

            streamClient = new Object();
            streamClient.onBWDone = onBWDone;

            connection = new NetConnection();
            connection.client = streamClient;
            connection.addEventListener(NetStatusEvent.NET_STATUS, monitorStatus);
            connection.connect("rtmp://localhost/live");
        }

        protected function monitorStatus(e:NetStatusEvent):void {
            traceField.appendText(e.info.code + "\n");
            if(e.info.code == "NetConnection.Connect.Success"){
                beginStreaming();
            }else if(e.info.code == "NetStream.Publish.Start"){
                traceField.appendText("\n" + e.info.description + "\n");
                traceField.appendText("codec: " + stream.videoStreamSettings.codec);
            }
        }

        protected function beginStreaming():void {
            var h264Settings:H264VideoStreamSettings = new H264VideoStreamSettings();
            h264Settings.setProfileLevel(H264Profile.BASELINE, H264Level.LEVEL_2);

            stream = new NetStream(connection);
            stream.addEventListener(NetStatusEvent.NET_STATUS, monitorStatus);
            stream.videoStreamSettings = h264Settings;
            stream.attachCamera(camera);
            stream.publish("mp4:h264livestream.f4v", "live");
        }

        public function onBWDone():void {}

    }
}
```

So long as we have Flash Media Server installed and running on our local machine, we will receive a message stating that the stream is being published and that the codec being used to encode the video is `H.264`, along with a preview of the camera feed. In this example, we are simply viewing the raw camera output and not the encoded stream.

www.it-ebooks.info

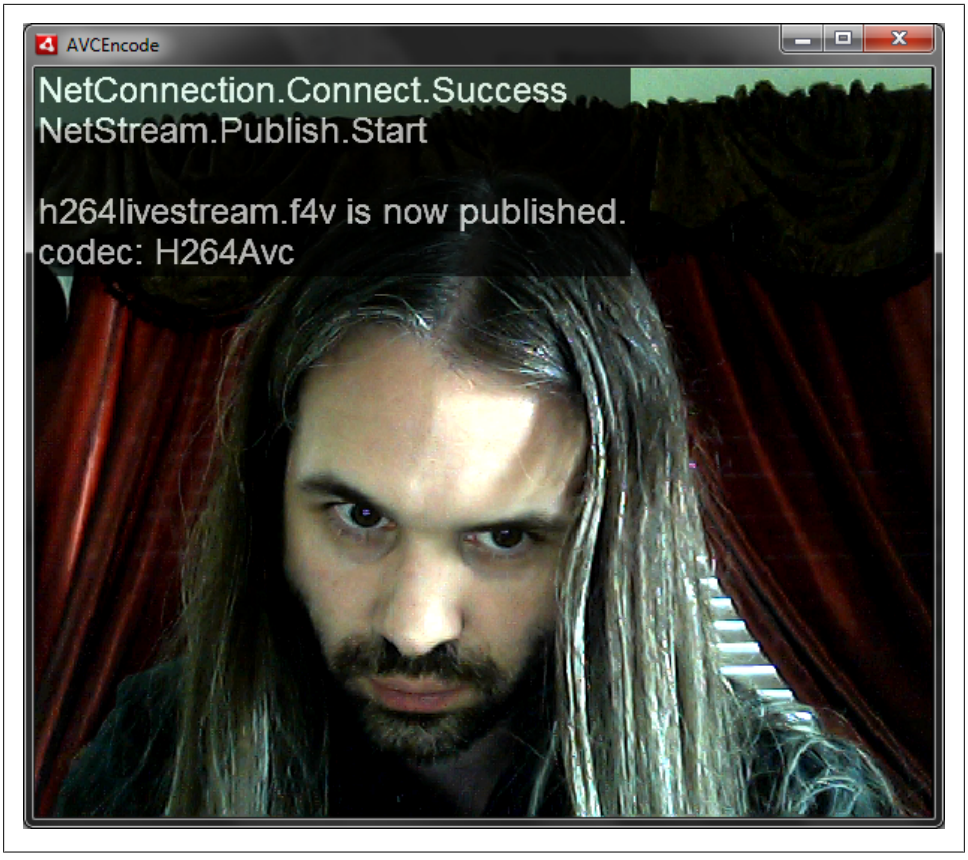


Figure 5-1. H.264 Encoding within AIR

Reading an H.264 Stream into AIR 3

The ability to decode `H.264` has been available since the initial release of AIR. To play back a stream or file encoded to `H.264` with AIR 3, we employ the same procedure we would normally use. First, create a `NetConnection` and establish a connection to Flash Media Server. In this case, we use *rtmp://localhost/live*, as the server exists on the same machine. We then hook in a `NetStream` object and request to play the previously published stream.

```
package {
    import flash.display.Sprite;
    import flash.events.NetStatusEvent;
    import flash.media.Video;
    import flash.net.NetConnection;
    import flash.net.NetStream;
    import flash.text.TextField;
    import flash.text.TextFormat;
```


www.it-ebooks.info

```
[SWF(width="600", height="500", backgroundColor="#CCCCCC")]

public class AVCPlayback extends Sprite {

    private var traceField:TextField;
    private var video:Video;
    private var connection:NetConnection;
    private var stream:NetStream;
    private var streamClient:Object;

    public function AVCPlayback() {
        generateDisplayObjects();
        performOperations();
    }

    protected function generateDisplayObjects():void {
        video = new Video(stage.stageWidth, stage.stageHeight);
        addChild(video);

        var defaultFormat:TextFormat = new TextFormat();
        defaultFormat.font = "Arial";
        defaultFormat.size = 24;
        defaultFormat.color = 0xFFFFFF;

        traceField = new TextField();
        traceField.backgroundColor = 0x000000;
        traceField.alpha = 0.7;
        traceField.autoSize = "left";
        traceField.background = true;
        traceField.defaultTextFormat = defaultFormat;
        addChild(traceField);
    }

    protected function performOperations():void {
        streamClient = new Object();
        streamClient.onBWDone = onBWDone;

        connection = new NetConnection();
        connection.client = streamClient;
        connection.addEventListener(NetStatusEvent.NET_STATUS, monitorStatus);
        connection.connect("rtmp://localhost/live");
    }

    protected function monitorStatus(e:NetStatusEvent):void {
        traceField.appendText(e.info.code + "\n");
        if(e.info.code == "NetConnection.Connect.Success"){
            beginStreaming();
        }
    }

    protected function beginStreaming():void {
        stream = new NetStream(connection);
        stream.addEventListener(NetStatusEvent.NET_STATUS, monitorStatus);
        stream.play("mp4:h264livestream.f4v");
```


www.it-ebooks.info

```
            video.attachNetStream(stream);
        }

        public function onBWDone():void {}
    }
}
```

When this class is compiled to *AIR*, so long as we have the encoded stream successfully published to Flash Media Server through the previous code example, we will see the published, natively-encoded `H.264` stream render through a `Video` object similar to Figure 5-2.

Figure 5-2. Decoded H.264 from Flash Player 11 encoded stream

G.711 Audio Compression for Telephony

AIR 3 includes support for the `G.711` codec for audio. `G.711` is actually rather old – actually formally standardized in 1972 as "Pulse Code Modulation (*PCM*) of voice

www.it-ebooks.info

frequencies". There are two compression algorithm variants of `G.711`, both of which can be used in AIR.

- **SoundCodec.PCMA** Specifies the G.711 A-law codec (used in Europe and the rest of the world).
- **SoundCodec.PCMU** Specifies the G.711 μ-law codec (used in North America and Japan).

`G.711` is primarily used in telephony and `SIP` (Session Initiation Protocol) based applications. It is tailored specifically for voice communications and supported on innumerable systems.

We can accomplish this through use of the `flash.media.SoundCodec` class within our audio project. Upon configuration of our `Microphone` object, we can assign the codec property to the `SoundCodec.PCMU` or `SoundCodec.PCMA` constant. This will ensure that any audio from that source is processed as `G.711`.

```
package {
    import flash.display.Bitmap;
    import flash.display.BitmapData;
    import flash.display.Sprite;
    import flash.events.SampleDataEvent;
    import flash.filters.BlurFilter;
    import flash.geom.Point;
    import flash.media.Microphone;
    import flash.media.SoundCodec;
    import flash.text.TextField;
    import flash.text.TextFormat;
    import flash.utils.ByteArray;

    [SWF(width="600", height="500", backgroundColor="#CCCCCC")]

    public class G711Telephony extends Sprite {

        private const SPECTRUM_COLOR:uint = 0x24ff00;

        private var traceField:TextField;
        private var microphone:Microphone;
        private var audioBlur:BlurFilter;
        private var audioBitmapData:BitmapData;
        private var audioBitmap:Bitmap;
        private var soundDisplay:Sprite;
        private var soundActivity:Sprite;

        public function G711Telephony() {
            generateTextFields();
            spectrumSetup();
            performOperations();
        }

        protected function generateTextFields():void {
```


www.it-ebooks.info

```
            var defaultFormat:TextFormat = new TextFormat();
            defaultFormat.font = "Arial";
            defaultFormat.size = 24;
            defaultFormat.color = 0xFFFFFF;

            traceField = new TextField();
            traceField.width = stage.stageWidth;
            traceField.height = stage.stageHeight;
            traceField.backgroundColor = 0x000000;
            traceField.alpha = 0.7;
            traceField.multiline = true;
            traceField.wordWrap = true;
            traceField.background = true;
            traceField.defaultTextFormat = defaultFormat;
            addChild(traceField);
        }

        protected function spectrumSetup():void {
           audioBitmapData = new BitmapData(stage.stageWidth, stage.stageWidth, true,
0x000000);
            audioBitmap = new Bitmap(audioBitmapData);

            audioBlur = new BlurFilter(2, 12, 2);

            soundActivity = new Sprite();
            soundDisplay = new Sprite();
            soundActivity.addChild(soundDisplay);
            soundActivity.addChild(audioBitmap);
            addChild(soundActivity);
        }

        protected function performOperations():void {
            if(Microphone.isSupported){
                microphone = Microphone.getMicrophone(0);
                microphone.codec = SoundCodec.PCMU;
                microphone.addEventListener(SampleDataEvent.SAMPLE_DATA,
microphoneDataSample);
            }
        }

        protected function microphoneDataSample(e:SampleDataEvent):void {
            var soundBytes:ByteArray = new ByteArray();
            soundBytes = e.data;

            traceField.text = "Microphone: " + e.target.name + "\n\n";
            traceField.appendText("activityLevel: " + e.target.activityLevel + "\n");
            traceField.appendText("bytesAvailable: " + soundBytes.bytesAvailable +
"\n");
            traceField.appendText("codec: " + e.target.codec);

            drawSpectrum(e.data);
        }

        protected function drawSpectrum(d:ByteArray):void {
            var ba:ByteArray = new ByteArray();
```


www.it-ebooks.info

```
            ba = d;
            var a:Number = 0;
            var n:Number = 0;
            var i:uint = 0;
            soundDisplay.graphics.clear();
            soundDisplay.graphics.lineStyle(2, SPECTRUM_COLOR, 0.8, false);
            soundDisplay.graphics.moveTo(0, (n/2)+(stage.stageHeight/2+100));
            for(i=0; i<=ba.bytesAvailable; i++) {
                a = ba.readFloat();
                n = a*soundActivity.height;
                soundDisplay.graphics.lineTo(i*(stage.stageWidth/ba.bytesAvailable),
(n/2)+(stage.stageHeight/2+100));
            }
            soundDisplay.graphics.endFill();
            audioBitmapData.draw(soundDisplay);
            audioBitmapData.applyFilter(audioBitmapData, audioBitmapData.rect, new
Point(0,0), audioBlur);
        }
    }
}
```

The result of running this class can be seen in Figure 5-3.

www.it-ebooks.info

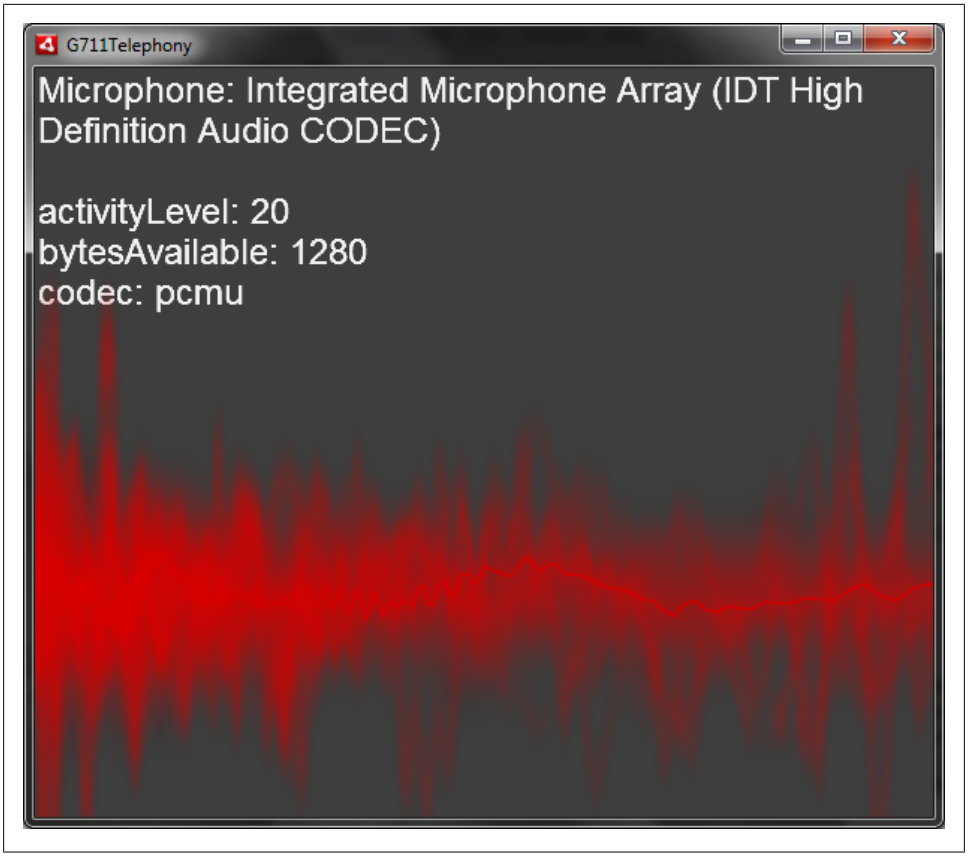


Figure 5-3. G.711 μ-law (PCMU) encoded audio data

www.it-ebooks.info

CHAPTER 6

Mobile Device Hardware Additions

One of the draws of developing for mobile devices is the array of input and output hardware that is built into these small units. Being able to tap into cameras, speakers, and microphones programmatically within our applications is not only desired functionality - but is something that is expected and necessary for many mobile applications. With AIR 3, developers receive a lot more control over these bits of hardware than in previous versions of the mobile runtime.

Camera Position API (Mobile)

Previously available on iOS and TabletOS platforms, the ability to access a front-facing device camera has now been extended to Android.

In order to differentiate between different cameras on a device, we can check the new `position` property of the `flash.media.Camera` class. There is also a new `flash.media.CameraPosition` class which specifies a set of constants that can be used when determining the current camera position.

`CameraPosition.FRONT` = front-facing device camera
`CameraPosition.BACK` = back-facing device camera
`CameraPosition.UNKNOWN` = indeterminate device camera

In the following example, we determine how many cameras are being reported on a device, and subsequently loop through each `Camera` object, testing against the `position` property to determine which one is the front-facing device camera. When the desired camera is located, we use that to draw to the screen.

```
package {
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.media.Camera;
    import flash.media.CameraPosition;
```


www.it-ebooks.info

```
        import flash.media.Video;

        [SWF(backgroundColor="#CCCCCC")]

        public class FrontCamera extends Sprite {

            private var video:Video;
            private var camera:Camera;

            public function FrontCamera() {
                super();
                stage.scaleMode = StageScaleMode.NO_SCALE;
                stage.align = StageAlign.TOP_LEFT;

                if(Camera.isSupported){
                    setupCamera();
                }
            }

            private function setupCamera():void {
                for(var i:int=0; i<Camera.names.length; ++i) {
                    camera = Camera.getCamera(String(i));
                    if (camera.position == CameraPosition.FRONT){
                        setupVideo();
                        break;
                    }
                }
            }

            private function setupVideo():void {
                video = new Video(stage.stageWidth, stage.stageHeight);
                video.attachCamera(camera);
                addChild(video);
            }
        }
    }
```

The front-facing camera will most likely be used to capture the user's face as seen in Figure 6-1.

www.it-ebooks.info

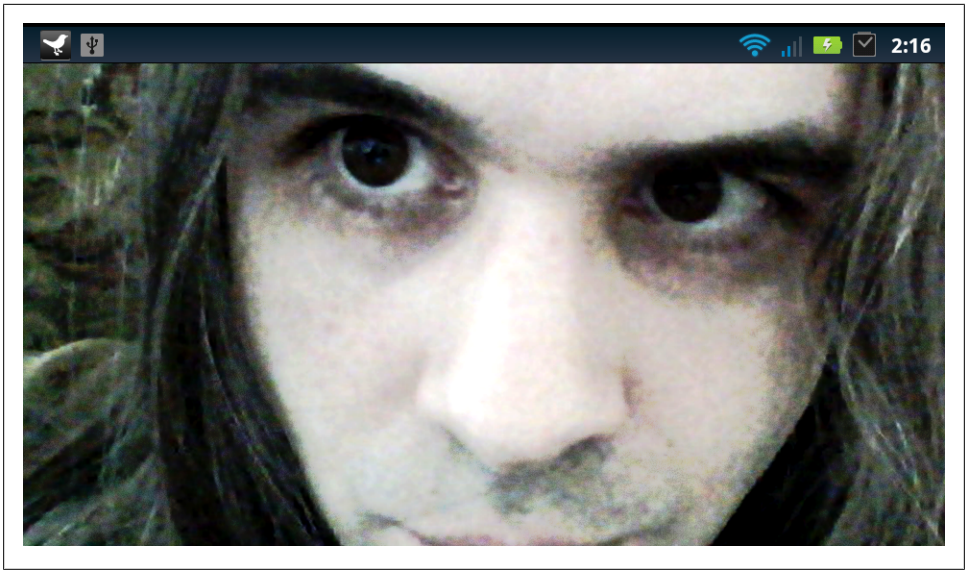
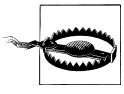


Figure 6-1. Front Facing Camera on Android

This is useful when a particular camera is needed within an AIR application, specifically in terms of video chat applications.



Be sure to provide proper permissions for the camera in your *application descriptor file* when targeting Android:

```
<uses-permission android:name="android.permission.CAMERA"/>
```

Device Speaker Control (Mobile)

Mobile devices generally have two separate hardware mechanisms for the transmission of audio. One of these is the speaker used in the transmission of voice data — primarily used for activities like phone calls. The other is a much fuller speaker, used for most any other activity: games, applications, music and video playback, et cetera.

Depending upon the needs of a particular application, AIR developers now have the ability to target a particular hardware speaker on the device. This is accessed by setting the `audioPlaybackMode` property of the `flash.media.SoundMixer` class to either the `AudioPlaybackMode.MEDIA` or `AudioPlaybackMode.VOICE` constants set through the `flash.media.AudioPlaybackMode` class.

Another interesting addition is the ability to override the default voice behavior through use of the `SoundMixer.useSpeakerphoneForVoice Boolean` property.

www.it-ebooks.info

```
package {
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.events.Event;
    import flash.media.AudioPlaybackMode;
    import flash.media.Sound;
    import flash.media.SoundMixer;
    import flash.net.URLRequest;
    import flash.text.TextField;
    import flash.text.TextFormat;

    [SWF(backgroundColor="#000000")]

    public class SoundSpeaker extends Sprite {

        private var traceField:TextField;
        private var sound:Sound;
        private var id3:Boolean;

        public function SoundSpeaker() {
            super();
            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            generateDisplayObjects();
            setupSoundMixer();
        }

        protected function generateDisplayObjects():void {
            var defaultFormat:TextFormat = new TextFormat();
            defaultFormat.font = "Arial";
            defaultFormat.size = 36;
            defaultFormat.color = 0xFFFFFF;

            traceField = new TextField();
            traceField.backgroundColor = 0x000000;
            traceField.alpha = 0.7;
            traceField.width = stage.stageWidth;
            traceField.height = stage.stageHeight;
            traceField.wordWrap = true;
            traceField.multiline = true;
            traceField.background = true;
            traceField.defaultTextFormat = defaultFormat;
            addChild(traceField);
        }

        private function setupSoundMixer():void {
            SoundMixer.audioPlaybackMode = AudioPlaybackMode.MEDIA;
            //SoundMixer.audioPlaybackMode = AudioPlaybackMode.VOICE;
            SoundMixer.useSpeakerphoneForVoice = false;

            setupSoundandLoad();
        }
```


www.it-ebooks.info

```
        private function setupSoundandLoad():void {
            sound = new Sound(new URLRequest("assets/drowning.mp3"));
            sound.addEventListener(Event.ID3, id3Loaded);
            sound.play();
        }

        protected function id3Loaded(event:Event):void {
            if(!id3){
                traceField.appendText("Playing: " + sound.id3.songName + "\n");
                traceField.appendText("From the album: " + sound.id3.album + "\n");
                traceField.appendText("By the artist: " + sound.id3.artist + "\n");
                traceField.appendText("Released in: " + sound.id3.year + "\n\n");
                traceField.appendText("Audio Playback Mode: " +
SoundMixer.audioPlaybackMode);
                id3 = true;
            }
        }
    }
}
```

The result of this code can be seen in Figure 6-2, running upon an Android device.

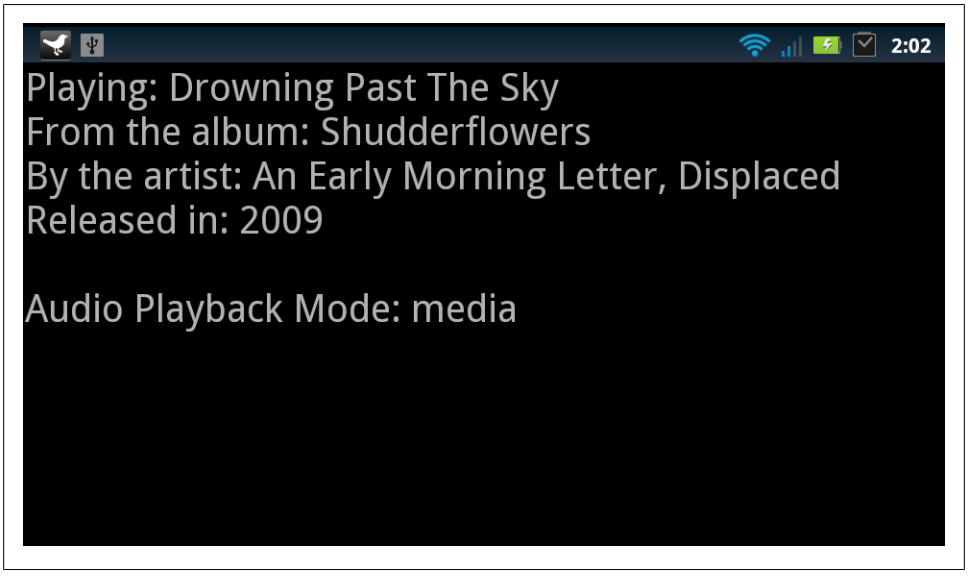

2:02
Playing: Drowning Past The Sky
From the album: Shudderflowers
By the artist: An Early Morning Letter, Displaced
Released in: 2009
Audio Playback Mode: media

Figure 6-2. Device speaker control

Background Audio Playback Support on iOS (Mobile)

With AIR 3, we are now able to inform iOS whether or not application audio should continue playing while the application is not in focus. We do this through manipulation of the AIR application descriptor file.

www.it-ebooks.info

Locate the `iPhone` node within your descriptor file and within the `InfoAdditions` node, insert a new `<key>` with a value of "UIBackgroundModes". Directly underneath this `<key>`, provide an `<array>` node with a `<string>` node nested within it. The value of this `<string>` node will be "audio".

That's all there is to it. Now compile to iOS as normal.

```
<!-- iOS specific capabilities -->
<iPhone>
    <InfoAdditions>
        <![CDATA[

            <key>UIDeviceFamily</key>
            <array>
                <string>1</string>
                <string>2</string>
            </array>

            <key>UIStatusBarStyle</key>
            <string>UIStatusBarStyleBlackOpaque</string>

            <key>UIRequiresPersistentWiFi</key>
            <string>YES</string>

            <key>UIBackgroundModes</key>
            <array>
                <string>audio</string>
            </array>

        ]]>
    </InfoAdditions>
    <requestedDisplayResolution>high</requestedDisplayResolution>
</iPhone>
```

This will enable any application audio to continue playback while our AIR for iOS mobile application runs in the background.

www.it-ebooks.info

CHAPTER 7

Data Transfer Additions

A robust application development platform needs a robust set of data transfer options. With the introduction of root level XML support along with the first release of Adobe AIR, developers could take advantage of any XML-based format within their applications. AIR 3 adds support for a native JSON handler and some great enhancements when working with sockets in communicating with various systems.

Native JSON (JavaScript Object Notation) Support

JavaScript Object Notation (*JSON*) is a hugely popular way of transporting structured data sets into and out of applications that run within AIR applications. Ever since ActionScript 3.0 was introduced, there have been third-party support libraries which allowed developers to use JSON in their projects quite easily; however, this is costly in terms of performance, since it was never a core function of AIR itself.

`JSON` is a top level class, similar to the `XML` or `Array` classes present in ActionScript. As such, they do not need to be imported in order to be used within an application.

The following `JSON` object describes a person through a series of name value pairs. Notice that objects can be nested and that this syntax can even include array structures. It is incredibly flexible.

```
{
     "firstName": "Joseph",
     "lastName": "Labrecque",
     "address":
     {
         "streetAddress": "2199 S. University Blvd.",
         "city": "Denver",
         "state": "CO",
         "postalCode": "80208"
     },
```


www.it-ebooks.info

```
        "phoneNumber":
        [
            {
              "type": "work",
              "number": "303.871.6566"
            },
            {
              "type": "fax",
              "number": "303.871.7445"
            }
        ]
}
```

JSON.parse()

We use this JSON file in the following code example to parse the values from the loaded `JSON` object using `JSON.parse()`, and then format the values within a basic `TextField` within our AIR Application

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.net.URLLoader;
    import flash.net.URLRequest;
    import flash.text.TextField;
    import flash.text.TextFormat;

    [SWF(width="600", height="500", backgroundColor="#CCCCCC")]

    public class ReadJSON extends Sprite {

        private var traceField:TextField;
        private var json:URLLoader;
        private var parsedJSON:Object;

        public function ReadJSON() {
            generateDisplayObjects();
            performOperations();
        }

        protected function generateDisplayObjects():void {
            var defaultFormat:TextFormat = new TextFormat();
            defaultFormat.font = "Arial";
            defaultFormat.size = 26;
            defaultFormat.color = 0xFFFFFF;

            traceField = new TextField();
            traceField.backgroundColor = 0x000000;
            traceField.alpha = 0.7;
            traceField.width = stage.stageWidth;
            traceField.height = stage.stageHeight;
            traceField.wordWrap = true;
            traceField.multiline = true;
            traceField.background = true;
```


www.it-ebooks.info

```
            traceField.defaultTextFormat = defaultFormat;
            addChild(traceField);
        }

        protected function performOperations():void {
            json = new URLLoader();
            json.addEventListener(Event.COMPLETE, parseJSON);
            json.load(new URLRequest("assets/data.json"));

            traceField.appendText("Loading JSON file...\n");
        }

        protected function parseJSON(e:Event):void {
            traceField.appendText("JSON file loaded successfully!\n");
            traceField.appendText("Parsing JSON...\n\n");
            traceField.appendText("RESULTS:\n");

            parsedJSON = JSON.parse(json.data);

            traceField.appendText("firstName: " + parsedJSON.firstName + "\n");
            traceField.appendText("lastName: " + parsedJSON.lastName + "\n");
            traceField.appendText("address.streetAddress: " +
parsedJSON.address.streetAddress + "\n");
            traceField.appendText("address.city: " + parsedJSON.address.city + "\n");
           traceField.appendText("address.state: " + parsedJSON.address.state + "\n");
            traceField.appendText("address.postalCode: " +
parsedJSON.address.postalCode + "\n");
            for(var i:int = 0; i<parsedJSON.phoneNumber.length; i++){
                traceField.appendText(parsedJSON.phoneNumber[i].type + ": " +
parsedJSON.phoneNumber[i].number + "\n");
            }
        }

    }
}
```

As you can see, dealing with JSON is very similar to dealing with XML within ActionScript. Parsing our imported `JSON` and outputting the data into a `TextField` renders similar to Figure 7-1.

www.it-ebooks.info

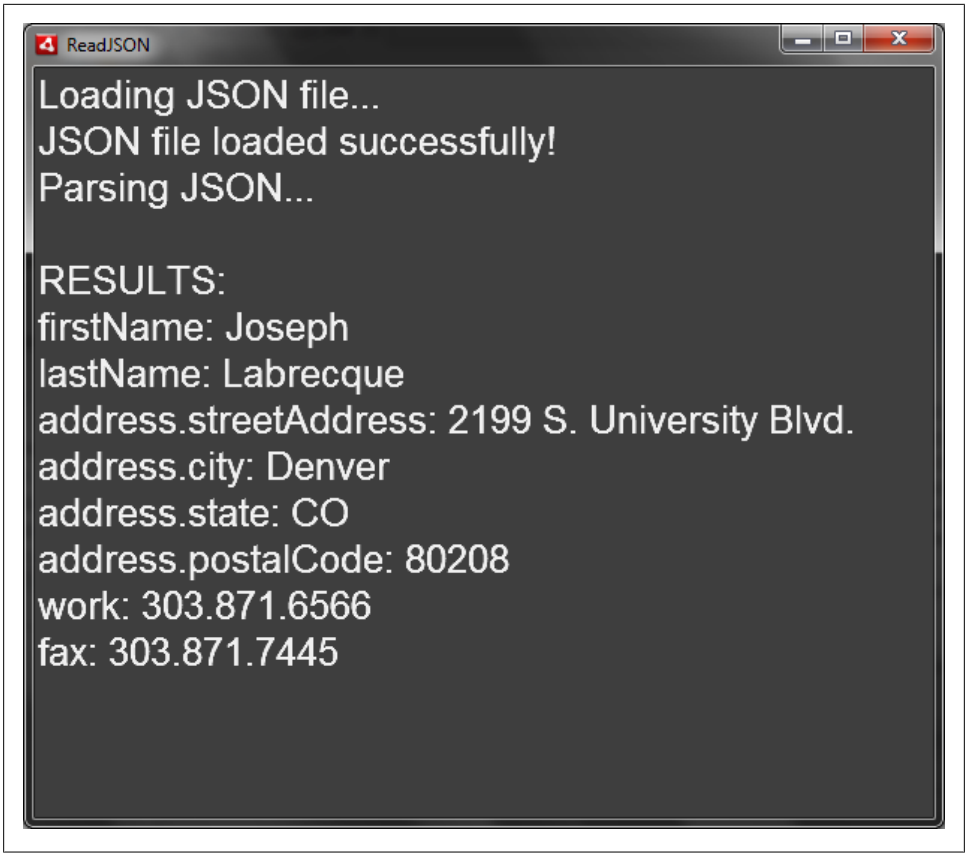


Figure 7-1. Parsed JSON output

JSON.stringify()

To actually write JSON from within an AIR project, we need to employ the `JSON.stringify()` method as demonstrated in the following code example.

```
package {
    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.text.TextFormat;

    [SWF(width="600", height="500", backgroundColor="#CCCCCC")]

    public class WriteJSON extends Sprite {

        private var traceField:TextField;
        private var jsonObject:Object;

        public function WriteJSON() {
```


www.it-ebooks.info

```
            generateDisplayObjects();
            performOperations();
        }

        protected function generateDisplayObjects():void {
            var defaultFormat:TextFormat = new TextFormat();
            defaultFormat.font = "Arial";
            defaultFormat.size = 26;
            defaultFormat.color = 0xFFFFFF;

            traceField = new TextField();
            traceField.backgroundColor = 0x000000;
            traceField.alpha = 0.7;
            traceField.width = stage.stageWidth;
            traceField.height = stage.stageHeight;
            traceField.wordWrap = true;
            traceField.multiline = true;
            traceField.background = true;
            traceField.defaultTextFormat = defaultFormat;
            addChild(traceField);
        }

        protected function performOperations():void {
            traceField.appendText("Forming Object in ActionScript...\n");

            jsonObject = new Object();
            jsonObject.firstName = "Edgar";
            jsonObject.middleName = "Allan";
            jsonObject.lastName = "Poe";
            jsonObject.birthDate = 1809;
            jsonObject.deathDate = 1849;
            jsonObject.nationality = "American";
            jsonObject.birthPlace = "Boston, Massachusetts";

            traceField.appendText("Stringify in progress...\n\n");

            var newJSON:Object = JSON.stringify(jsonObject, null, 4);

            traceField.appendText("RESULT:\n");
            traceField.appendText(newJSON.toString());
        }

    }
}
```

The `JSON.stringify()` method will convert the ActionScript object we've assembled into complete JSON syntax. This assumes that all data types within the object are of a type that can be converted into JSON.

Valid data types include: `Array`, `String`, `Number`, `Boolean`, and `null`.

www.it-ebooks.info

The first argument that we pass into this method is the actual object which we want to convert. The second argument is an optional replacer function (or array) that can be used to transform or filter the key/value pairs in the object to be converted. This could be used in case we want to exclude certain key/value pairs, for instance, from the actual output.

The final argument specifies the amount of spaces to insert before each piece of data in order to make it more human-readable. In this example, we are passing in the number 4, specifying that the method should prepend 4 whitespace characters before each entry. This is how we achieve the spacing and readability in Figure 7-2.

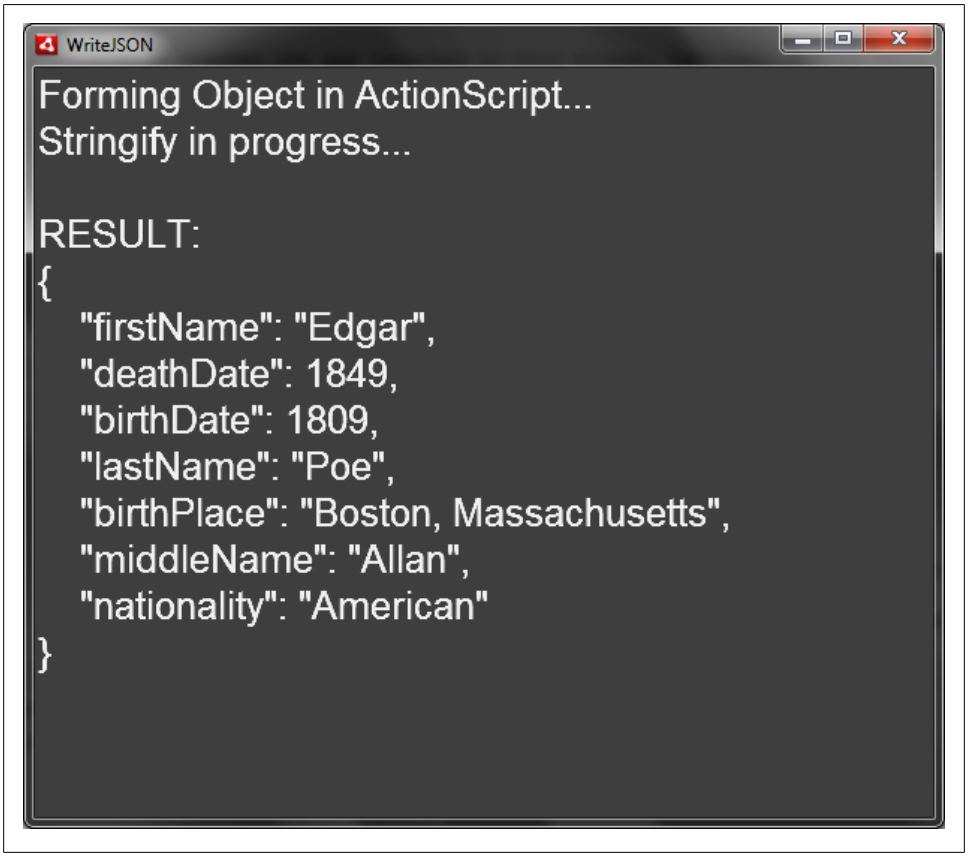


Figure 7-2. Stringified object output

Figure 7-2 demonstrates the stringified output from our example.

www.it-ebooks.info

Socket Progress Events

The `flash.net.Socket` class has been available to us since the initial release of Adobe AIR. When using sockets in ActionScript, developers have always had access to an event to monitor the progress of input data coming through the socket by using `flash.events.ProgressEvent`. Up until now, however, monitoring the output data being sent across a socket connection has been nearly impossible.

With AIR 3, we now have access to the `flash.events.OutputProgressEvent` class, with which we can easily monitor both the pending and total bytes being sent out over a socket connection. This can be used in order to display something like a progress indicator to the user, sequence certain events within an application, or simply verify that the data has been fully processed over the socket.

In the following example, we establish a socket connection and monitor both the `flash.events.ProgressEvent` and `flash.events.OutputProgressEvent` events when connected to adobe.com using a basic socket connection.

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.IOErrorEvent;
    import flash.events.OutputProgressEvent;
    import flash.events.ProgressEvent;
    import flash.net.Socket;
    import flash.text.TextField;
    import flash.text.TextFormat;
    import flash.utils.ByteArray;

    [SWF(width="600", height="500", backgroundColor="#CCCCCC")]

    public class SocketProgress extends Sprite {

        private var traceField:TextField;
        private var socket:Socket;

        public function SocketProgress() {
            generateDisplayObjects();
            performOperations();
        }

        protected function generateDisplayObjects():void {
            var defaultFormat:TextFormat = new TextFormat();
            defaultFormat.font = "Arial";
            defaultFormat.size = 22;
            defaultFormat.color = 0xFFFFFF;

            traceField = new TextField();
            traceField.backgroundColor = 0x000000;
            traceField.alpha = 0.7;
            traceField.width = stage.stageWidth;
            traceField.height = stage.stageHeight;
            traceField.wordWrap = true;
```


www.it-ebooks.info

```
            traceField.multiline = true;
            traceField.background = true;
            traceField.defaultTextFormat = defaultFormat;
            addChild(traceField);
        }

        protected function performOperations():void {
            socket = new Socket();
            socket.addEventListener(Event.CONNECT, socketConnected);
            socket.addEventListener(IOErrorEvent.IO_ERROR, socketError);
            socket.addEventListener(ProgressEvent.SOCKET_DATA, socketProgress);

            socket.addEventListener(OutputProgressEvent.OUTPUT_PROGRESS,
socketOutputProgress);

            socket.connect("adobe.com", 80);
            traceField.text = "Attempting Connection...\n\n";
        }

        protected function socketProgress(e:ProgressEvent):void {
            var byteArray:ByteArray = new ByteArray();
            traceField.appendText("SOCKET DATA RETURNED:\n");
            socket.readBytes(byteArray, 0, socket.bytesAvailable);
            traceField.appendText(byteArray.toString());
        }

        protected function socketOutputProgress(e:OutputProgressEvent):void {
           traceField.appendText("OUTPUT PROGRESS => bytesPending: " + e.bytesPending
+ " / bytesTotal: " + e.bytesTotal + "\n\n");
        }

        protected function socketConnected(e:Event):void {
            traceField.appendText("Socket Connected!\n\n");
            socket.writeUTFBytes("GET/HTTP/1.1\nHost: adobe.com");
        }


        protected function socketError(e:IOErrorEvent):void {
            traceField.appendText("IO Error: " + e.text + "\n\n");
        }

    }
}
```

The result of the above code is shown in Figure 7-3. As we can see, we now have the ability to monitor the bytes being returned over our established socket as well as access the total bytes to expect from this transaction.

www.it-ebooks.info

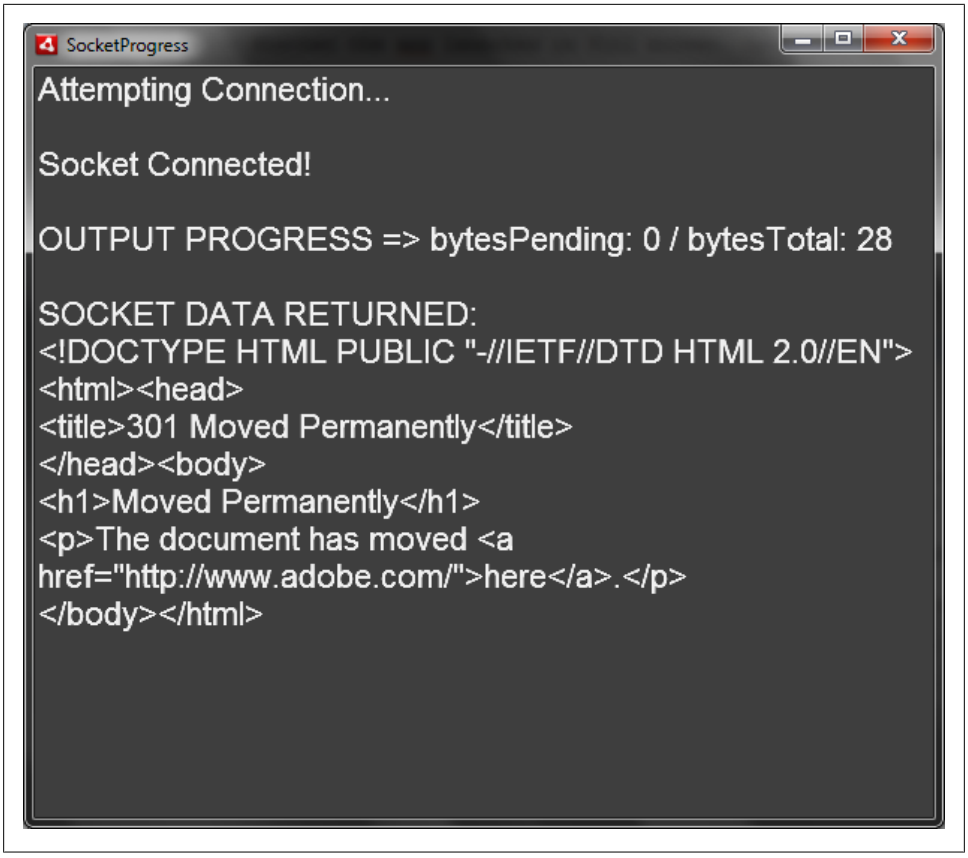


Figure 7-3. Socket output progress

www.it-ebooks.info

www.it-ebooks.info

CHAPTER 8

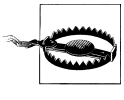
Runtime Enhancements

Among the language and runtime enhancements in AIR 3 are a variety of new classes, methods, properties, and architectures whose aim it is to make things easier, extensible, and faster in regard to the runtime and its usage. In this chapter, we will have a look at the variety of language and runtime improvements along with general implementation examples that can be easily built upon for a variety of projects.

ActionScript Native Extensions

With the release of AIR 2, developers were given the option in desktop AIR to interface with native processes on the host operating system. This was great, as it meant that an AIR application could be somewhat extended by passing messages back and forth between the application and a running process. It wasn't ideal though, and wasn't available for mobile AIR projects.

Now, with AIR 3 we have something much more powerful at our disposal: ActionScript Native Extensions (*ANE*). This new functionality allows a developer to write the bulk of an application in ActionScript but also provide a portion of it in the language of the native operating system in order to extend the AIR runtime itself, providing hooks into hitherto untouchable possibilities.



Note that the bulk of the code will more than likely be written in a language other than ActionScript!

For example, while the AIR runtime has native support for sensors like the accelerometer on iOS or Android...it does not provide any way of tapping into a gyroscope, barometer, or vibration component of the device, which may be present. Likewise, core OS data and functionality such as the device contact list or the ability to transmit notifications remain completely hidden from AIR. With *ANEs*, a developer can write the code to extend the capabilities of AIR in order to access such things within the application.

www.it-ebooks.info

While we are focusing on using ActionScript Native Extensions for mobile in this chapter, note that *ANE* can be used on desktop as well!

In this example, we will use a Vibrator *ANE* developed for use on Android. An *ANE* is similar to a *SWC* and is included in our project through a very familiar method. In this example, we'll use Flash Builder 4.6 to include the Vibrator *ANE* into an AIR for Android project. Choose the project you are working on and click *File* > *Properties*. This will open up the Properties dialog in which you will choose *ActionScript Build Path* (alternatively: *Flex Build Path*) on the left and choose the *Native Extensions* tab. Here you may click the button labeled *Add ANE…*and simply browse to the *ANE* file you wish to build into your project.

You can also twirl down elements of the *ANE* package to view target data and *ANE* limitations.

www.it-ebooks.info

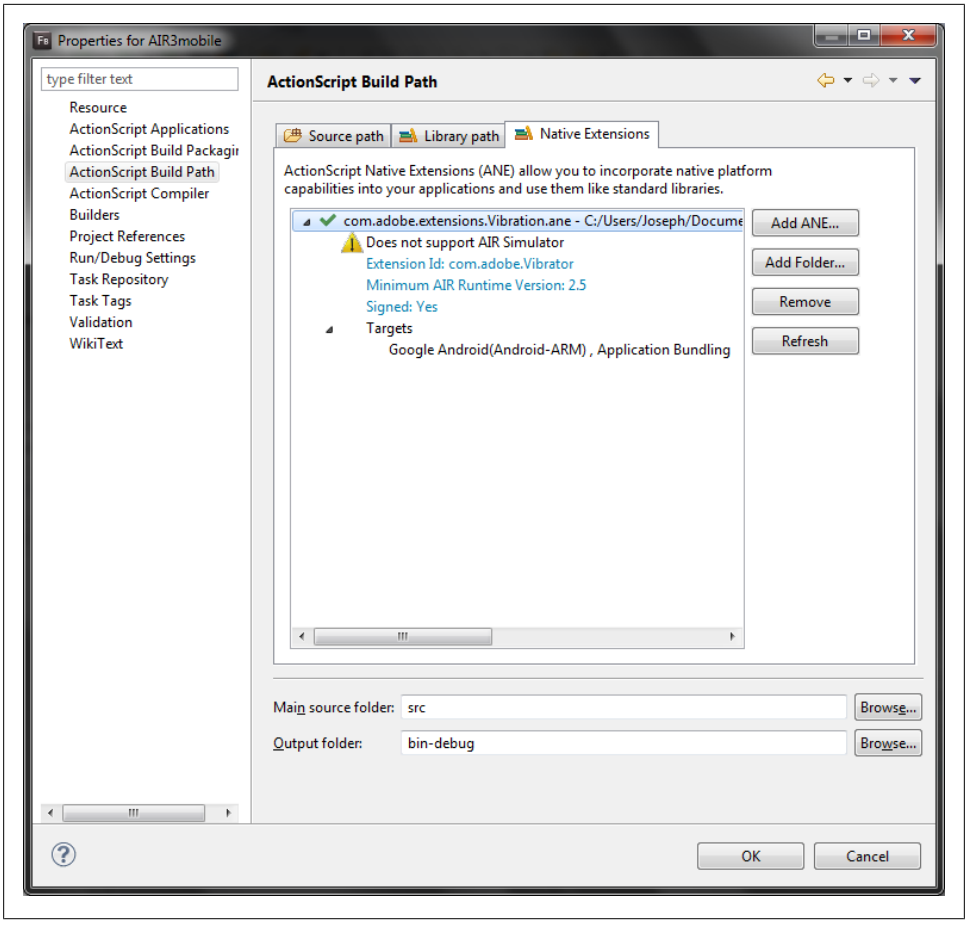


Figure 8-1. Vibrator ANE added to mobile project

Next, we will need to be sure and package the *ANE* into our build settings. With the *Properties* window still open, click *ActionScript Build Packaging* (alternatively: *Flex Build Packaging*) on the left and choose the *Native Extensions* tab. Here, we simply select which *ANE* files we want to include with our build for each platform. This is a major difference between *ANE* and *SWC* in that *ANE* must be packaged separately in this manner.

www.it-ebooks.info

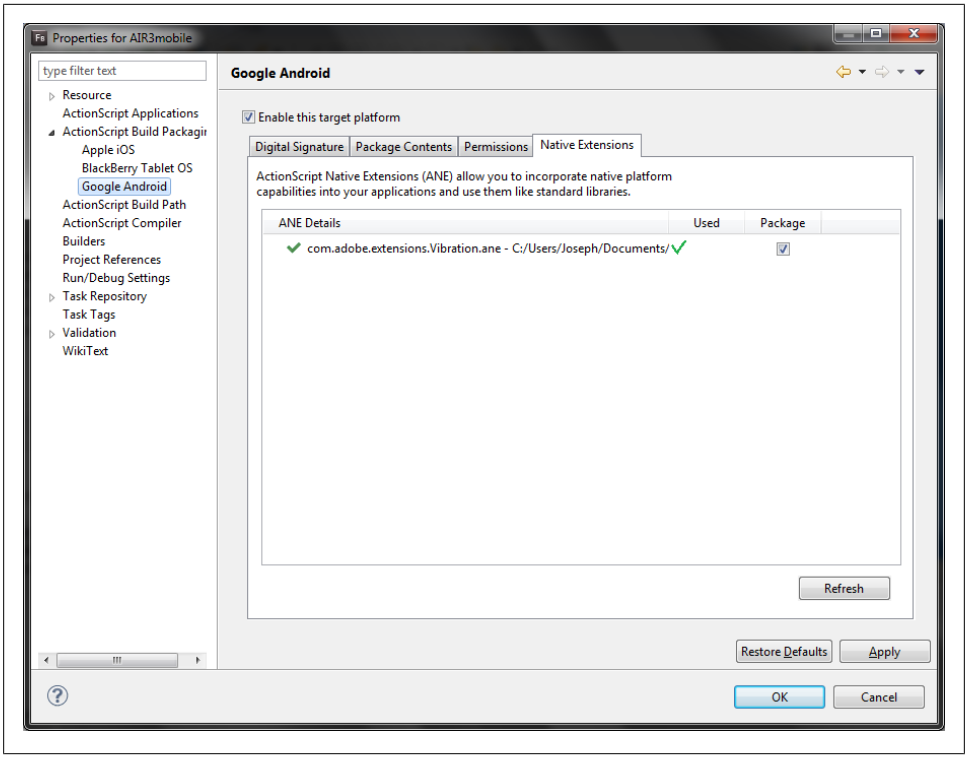


Figure 8-2. Including an ANE into the build package

When using an *ANE* on Android, we may also need to include various permissions within the `<android>` node of our application descriptor file. Here, we add permissions for vibrate.

```
<android>
    <manifestAdditions><![CDATA[
        <manifest android:installLocation="auto">
            <uses-permission android:name="android.permission.INTERNET"/>
            <uses-permission android:name="android.permission.VIBRATE"/>
        </manifest>
    ]]></manifestAdditions>
</android>
```

For any extensions we add to a project, we must declare them within the application descriptor file root as demonstrated here. Flash Builder 4.6 will actually perform this action for us, if desired.

```
<extensions>
    <extensionID>com.adobe.Vibrator</extensionID>
</extensions>
```


www.it-ebooks.info

Does this sound like a lot of preparation? It is...but it only needs to be done once per project, and considering the power available through ActionScript Native Extensions, it is well worth this small effort.

To effectively use this Vibrator *ANE* within our project, we must first import the `Vibrator` class. We can then instantiate a new `Vibrator` object and invoke `Vibrator.vibrate()` whenever we wish to cause our device to vibrate. Passing in a certain number of milliseconds as an argument will instruct the device to vibrate for that specific length of time. In the example below, we cause the device to vibrate based upon the user touching the `Stage`, and display a message along with the current reading from the `flash.utils.getTimer` class.

```
package {
    import com.adobe.nativeExtensions.Vibrator.Vibrator;

    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.events.MouseEvent;
    import flash.text.TextField;
    import flash.text.TextFormat;
    import flash.utils.getTimer;

    [SWF(backgroundColor="#000000")]

    public class MobileVibrate extends Sprite {

        private var traceField:TextField;
        private var vibrator:Vibrator;

        public function MobileVibrate() {
            super();
            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            generateDisplayObjects();
            performOperations();
        }

        protected function generateDisplayObjects():void {
            var defaultFormat:TextFormat = new TextFormat();
            defaultFormat.font = "Arial";
            defaultFormat.size = 36;
            defaultFormat.color = 0xFFFFFF;

            traceField = new TextField();
            traceField.backgroundColor = 0x000000;
            traceField.alpha = 0.7;
            traceField.width = stage.stageWidth;
            traceField.height = stage.stageHeight;
            traceField.wordWrap = true;
            traceField.multiline = true;
```


www.it-ebooks.info

```
            traceField.background = true;
            traceField.defaultTextFormat = defaultFormat;
            addChild(traceField);
        }

        protected function performOperations():void {
            if(Vibrator.isSupported){
                vibrator = new Vibrator();
                stage.addEventListener(MouseEvent.CLICK, performVibration);
                traceField.appendText("Click anywhere to vibrate device...\n\n");
            }else{
                traceField.appendText("Vibrator not supported!!!");
            }
        }

        protected function performVibration(e:MouseEvent):void {
            vibrator.vibrate(1000);
            traceField.appendText("VIBRATE! [" + getTimer() + "]\n");
        }

    }
}
```

When this code is compiled to *APK* and run on Android, it will appear similar to Figure 8-3.

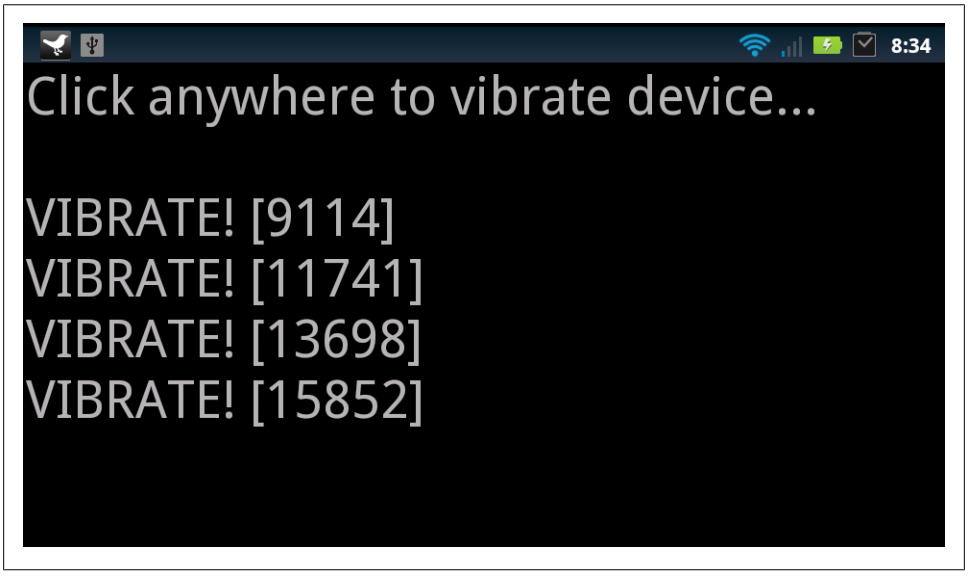


Figure 8-3. Vibration ANE on Android

www.it-ebooks.info

Through the use of ActionScript Native Extensions (*ANE*), developers are also able to leverage such things as the **Android Marketing Licensing Service** in order to intelligently enforce licensing policies for applications published through the **Android Market**.

There are many *ANE* files already available from Adobe and the Flash Platform Community that are freely available for developers to use within their applications. For those individuals comfortable with writing native code, the process to create and package *ANE* files is heavily documented by Oliver Goldman over at *http://www.adobe.com/devnet/air/articles/extending-air.html*

Captive Runtime Support

If there has been any feature that is most often requested for AIR, this is probably it (if not, it is certainly up there). Captive Runtime will package an application in such a way that it does not require a separate AIR runtime to be installed on a user's system. This is conceptually similar to how AIR is able to function on iOS — the AIR runtime is packaged along with the application to form a complete package.

Captive Runtime is now available on Windows, OSX, and Android. When packaging AIR for iOS, the process has always bundled the AIR runtime, due to iOS restrictions against external runtimes.

One of the benefits of embedding the AIR runtime along with your application is obvious: the user will not be troubled with an addition download and install if the desktop or device does not already have AIR installed. Another benefit is for those users on tightly controlled networks that do not allow extra runtimes to be installed upon corporate machines. There are drawbacks to this approach, however: embedding the runtime will cause an increase in file size for the distributable package. This can be anywhere from 5–8 megabytes depending upon the platform. A developer must weight both the good and the bad when deciding which distribution method is taken for the runtime — but such choices are always better than no choice at all.

To package an application using this feature on the desktop with Flash Builder 4.6, choose *Project > Export Release Build*. Be sure that *Signed application with captive runtime* is selected and hit *Next >*.

www.it-ebooks.info

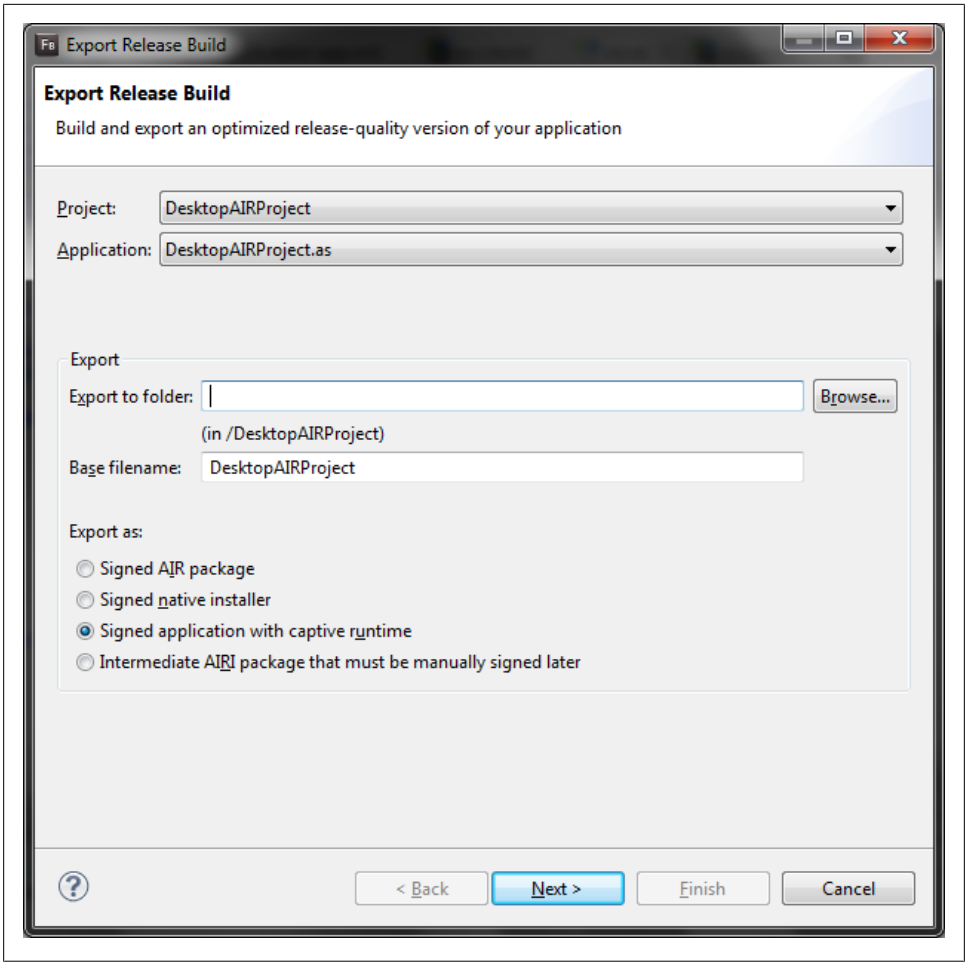


Figure 8-4. AIR desktop captive runtime

To embed the AIR runtime within an Android application with Flash Builder 4.6, choose *Project* > *Export Release Build*. Be sure that *Signed packages for each target platform* is selected and hit *Next* >.

www.it-ebooks.info

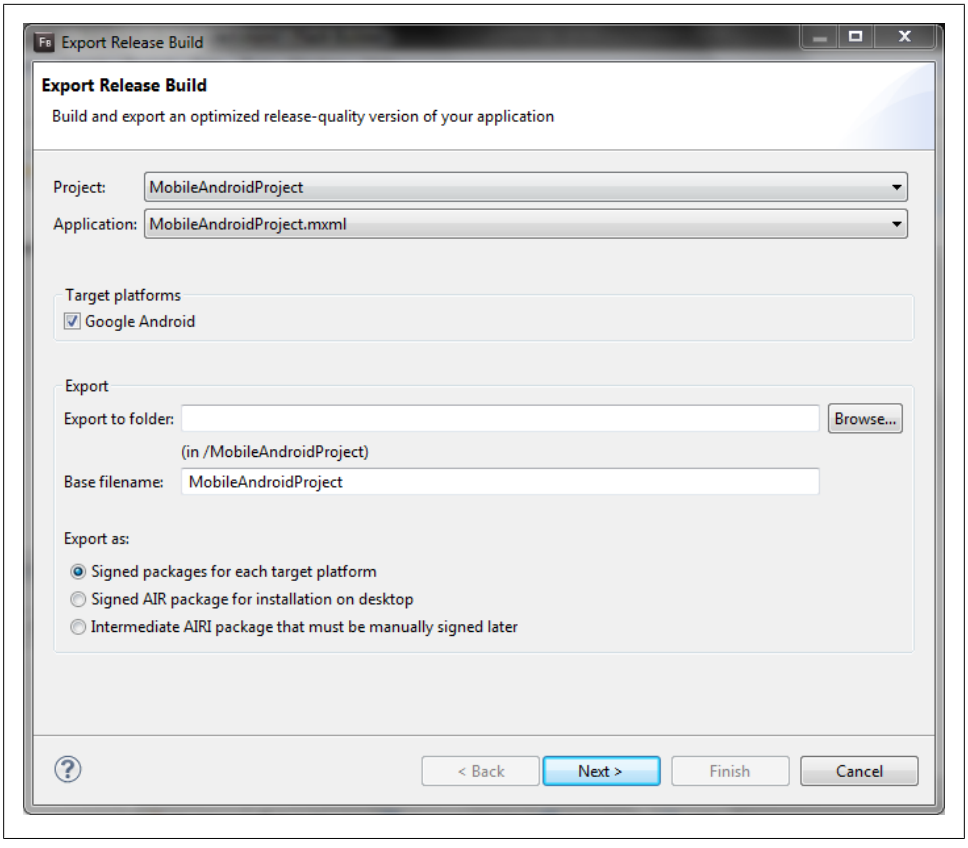


Figure 8-5. Export Android project

Once you reach the Packaging Settings window for Android, be sure to select the *Deployment* tab and choose *Export application with embedded AIR runtime* in order to package and build an *APK* with the embedded runtime.

www.it-ebooks.info

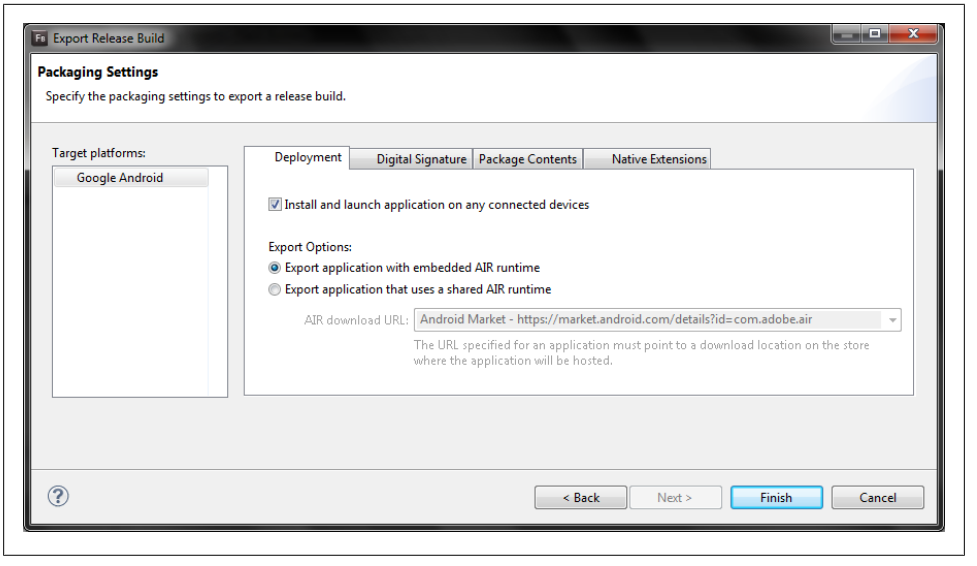
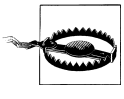

Export Release Build
Packaging Settings
Specify the packaging settings to export a release build.
Target platforms:
Google Android
Deployment
Digital Signature
Package Contents
Native Extensions
Install and launch application on any connected devices
Export Options:
Export application with embedded AIR runtime
Export application that uses a shared AIR runtime
AIR download URL:
Android Market - https://market.android.com/details?id=com.adobe.air
The URL specified for an application must point to a download location on the store where the application will be hosted.
< Back
Next >
Finish
Cancel

Figure 8-6. Android embedded runtime

It is also possible to manually package an Android or desktop application from the command line with the embedded runtime option.

Android Color Depth Setting (Mobile)

When targeting AIR 3 for Android, we can now specify whether we want the application compiled for either a 16-bit or 32-bit color mode. To accomplish this, we will need to modify the AIR application descriptor file.



This will only work with `<renderMode>` set to "cpu" or "auto". The setting is not effective in "gpu" mode since it requires a 32-bit color mode.

To specify a certain color space, be sure to first locate the `<initialWindow>` node and find the `<renderMode>` node which is nested within. Be sure that the value of `<renderMode>` is set to "auto" or "cpu".

```
<android>
    <manifestAdditions>
        <![CDATA[

            <manifest android:installLocation="auto">

                <uses-permission android:name="android.permission.INTERNET"/>
```


www.it-ebooks.info

```
                <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
              <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/
>
                <uses-feature android:required="true"
android:name="android.hardware.touchscreen.multitouch"/>

                <application android:enabled="true">
                    <activity android:excludeFromRecents="false">
                        <intent-filter>
                            <action android:name="android.intent.action.MAIN"/>
                          <category android:name="android.intent.category.LAUNCHER"/>
                        </intent-filter>
                    </activity>
                </application>

            </manifest>

        ]]>
    </manifestAdditions>

    <colorDepth>32bit</colorDepth>

</android>
```

Using this new manifest attribute, we can specify an appropriate color space for our AIR for Android mobile application.

Note that if left unspecified, this setting will default to 16-bit color for AIR 2.7 and older, whereas AIR 3 and newer will be compiled to take advantage of 32-bit color depth.

Garbage Collection Advice

Within the `flash.system.System` class is a new method called `pauseForGCIfCollectionImminent()`. The method accepts a single argument that determines the desired imminence value. This optional value must be a `Number` between 0 and 1, where lower values indicate a less intense need for a garbage collection sweep to occur. Using this method, a developer can advise the garbage collector to begin performing its tasks at a time when it is convenient to do so.

In the example below, we create a small "game" with two states. One of these states represents a play level which the user would interact with and is a timed level. The second state represents the paused time between levels, in which the user is provided with a chance to reflect upon their achievements and prepare for another level of play. With `System.pauseForGCIfCollectionImminent()` invoked during the proper time, a developer can advise the garbage collector to perform its actions at the best possible time.

```
package {
    import flash.display.Sprite;
```


www.it-ebooks.info

```
import flash.events.Event;
import flash.events.TimerEvent;
import flash.system.System;
import flash.text.TextField;
import flash.text.TextFormat;
import flash.utils.Timer;

[SWF(width="600", height="500", backgroundColor="#CCCCCC")]

public class GCAdvice extends Sprite {

    private var traceField:TextField;
    private var stateName:String;
    private var levelTimer:Timer;
    private var msg:String;

    public function GCAdvice() {
        generateDisplayObjects();
        performOperations();
    }

    protected function generateDisplayObjects():void {
        var defaultFormat:TextFormat = new TextFormat();
        defaultFormat.font = "Arial";
        defaultFormat.size = 26;
        defaultFormat.color = 0xFFFFFF;

        traceField = new TextField();
        traceField.backgroundColor = 0x000000;
        traceField.alpha = 0.7;
        traceField.width = stage.stageWidth;
        traceField.height = stage.stageHeight;
        traceField.wordWrap = true;
        traceField.multiline = true;
        traceField.background = true;
        traceField.defaultTextFormat = defaultFormat;
        addChild(traceField);
    }

    protected function performOperations():void {
        stateName = "GamePlaying";
       msg = "\n\nUser is playing the game and we do not want to be disruptive by
advising the GC at this time...";
      msg += "\n\nIf we were to try and invoke the GC while a user is interacting
with our game, and while a number of CPU intensive processes are in play, it can
actually freeze the game temporarily.";
        msg += "\n\nThis could cause the user to fail at his task and is visually
disruptive.";

        stage.addEventListener(Event.ENTER_FRAME, monitorGameState);
        levelTimer = new Timer(5000, 1);
        levelTimer.addEventListener(TimerEvent.TIMER_COMPLETE, timeUp);
        levelTimer.start();
    }
```


www.it-ebooks.info

```
        protected function monitorGameState(e:Event):void {
            traceField.text = "Current State: " + stateName;
            traceField.appendText(msg);
        }

        protected function timeUp(e:TimerEvent):void {
            stateName = "LevelComplete";

            System.pauseForGCIfCollectionImminent();

            msg = "\n\nSystem.pauseForGCIfCollectionImminent() invoked!";
            msg += "\n\nWe do this between levels to avoid any strange behavior such
as display glitches due to the GC running.";
            msg += "\n\nBy advising the GC to fire during a time of little user
interaction and game engine processes - we reduce the risk of disruption
considerably.";
            msg += "\n\nConvenient!";
        }

    }
}
```

During gameplay, we actually do not want to do anything to cause the garbage collector to fire, because this could cause the game to freeze up for a few seconds while the process is completed. It is much better to run the garbage collector during some other time in the game lifecycle, preferably when there is little going on and the user is less engaged with the screen, such as in the case of a level complete or level loading screen. The important thing being that there is no disruption to the user experience.

In our demonstration code, we include a `Timer` to decide when to move from a state of active engagement and into a state of rest between levels. This state of rest is the perfect time to advise the garbage collector to perform its duties through `System. pauseForGCIfCollectionImminent()`.

www.it-ebooks.info

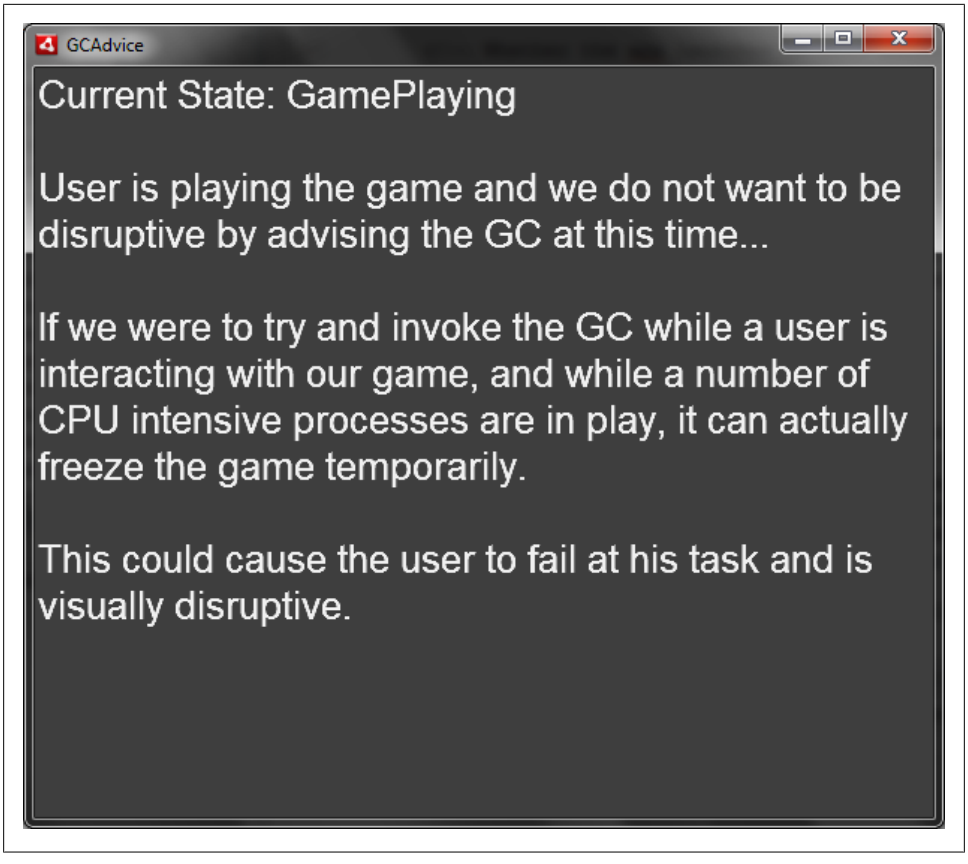

GCAdvice
Current State: GamePlaying
User is playing the game and we do not want to be disruptive by advising the GC at this time...
If we were to try and invoke the GC while a user is interacting with our game, and while a number of CPU intensive processes are in play, it can actually freeze the game temporarily.
This could cause the user to fail at his task and is visually disruptive.

Figure 8-7. Example SWF during the "GamePlaying" state

Once the `Timer` completes for a level, the game will enter a "LevelComplete" state in which the user is able to pause for a moment before continuing on to the next level of game play. This is the perfect time to perform any garbage collection, since the user is no longer actively engaged and there is basically nothing happening on screen. If the garbage collector were to cause any sort of freezing or similar visual disruption, the user would in all likelihood never even notice.

www.it-ebooks.info

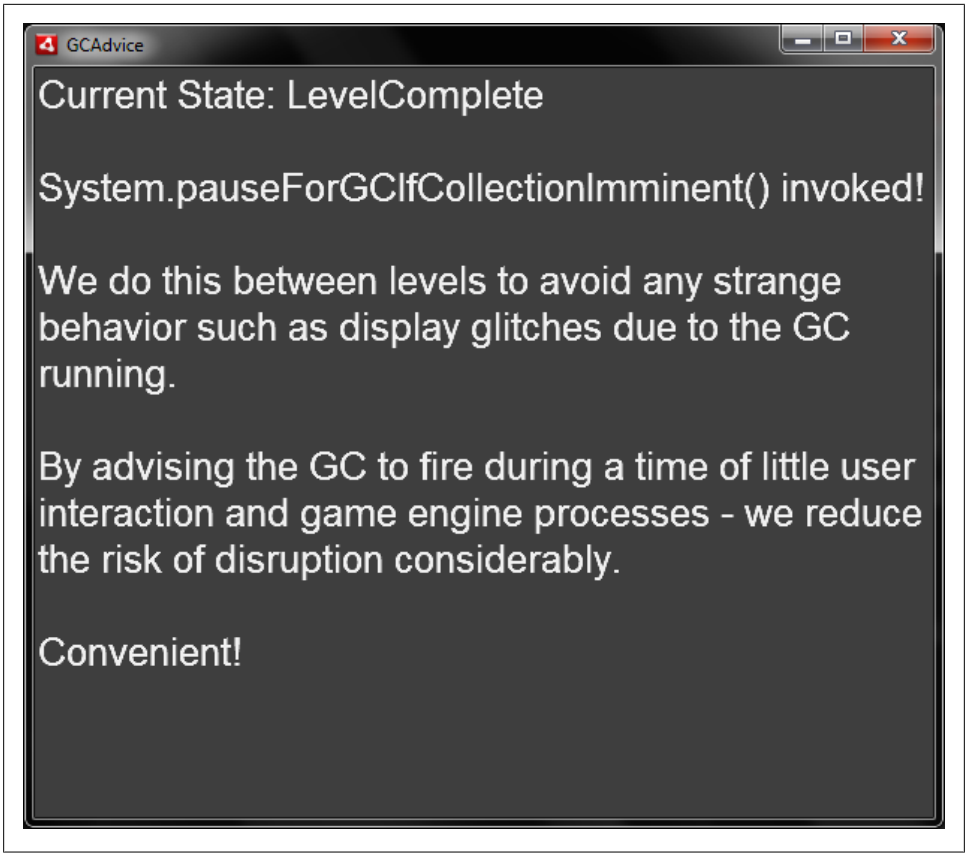
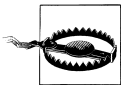

GCAdvice
Current State: LevelComplete
System.pauseForGCIfCollectionImminent() invoked!
We do this between levels to avoid any strange behavior such as display glitches due to the GC running.
By advising the GC to fire during a time of little user interaction and game engine processes - we reduce the risk of disruption considerably.
Convenient!

Figure 8-8. Example SWF during the "LevelComplete" state



It's important to note that while a developer can choose when to advise the garbage collector to run within a game or application by invoking `System. pauseForGCIfCollectionImminent()`, there is no guarantee that it will actually fire at that time.

www.it-ebooks.info

www.it-ebooks.info

CHAPTER 9

Adobe AIR Security

Security in any environment is always a just concern. As platforms expand and the general technology landscape shifts, new problems will crop up that require attention and additional mechanisms will be put into place in order to harden platform defenses. AIR 3 includes a number of new and updated security mechanisms, including new APIs for secure data generation along with expanded runtime support for streaming protected video for desktop and mobile.

Encrypted Local Storage (Mobile)

Encrypted local storage (*ELS*) has been available on the desktop in AIR from the first release, but has not been available for use with mobile platforms. With AIR 3, we can now leverage the `flash.data.EncryptedLocalStore` class on mobile devices as well.

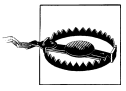
On Android, the data is protected by Android's user ID-based file system security.

To use `EncryptedLocalStore` on a mobile device, we will first check and see whether it is supported by checking against the `EncryptedLocalStore.isSupported` Boolean property. If this returns `true`, we can then perform both `EncryptedLocalStore.setItem()` and `EncryptedLocalStore.getItem()` commands within our application.

To set items to the *ELS* using `EncryptedLocalStore.setItem()`, we must pass in at least two arguments: a name to identify the data, and the data itself. The data will need to be encoded into a `ByteArray` before it is set to the *ELS* through `EncryptedLocalStore.getItem()`. To retrieve the data, we simply pass in the name that we had given it previously.

www.it-ebooks.info

To remove certain items from the *ELS*, we can invoke `EncryptedLocalStore.removeItem()`, passing in the given name as an identifier. Invoking `EncryptedLocalStore.reset()` will clear all data, wiping it completely from the system.



When an application is removed, the uninstaller does not delete data stored in the encrypted local store.

```
package {
    import flash.data.EncryptedLocalStore;
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.text.TextField;
    import flash.text.TextFormat;
    import flash.utils.ByteArray;

    [SWF(backgroundColor="#000000")]

    public class MobileEncryptedStore extends Sprite {

        private var traceField:TextField;

        public function MobileEncryptedStore() {
            super();
            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            generateDisplayObjects();
            performOperations();
        }

        protected function generateDisplayObjects():void {
            var defaultFormat:TextFormat = new TextFormat();
            defaultFormat.font = "Arial";
            defaultFormat.size = 36;
            defaultFormat.color = 0xFFFFFF;

            traceField = new TextField();
            traceField.backgroundColor = 0x000000;
            traceField.alpha = 0.7;
            traceField.width = stage.stageWidth;
            traceField.height = stage.stageHeight;
            traceField.wordWrap = true;
            traceField.multiline = true;
            traceField.background = true;
            traceField.defaultTextFormat = defaultFormat;
            addChild(traceField);
        }

        protected function performOperations():void {
            traceField.text = "EncryptedLocalStore.isSupported => " +
EncryptedLocalStore.isSupported + "\n\n";
```


www.it-ebooks.info

```
            if(EncryptedLocalStore.isSupported){
                writeEncryptedLocalStore();
            }
        }

        protected function writeEncryptedLocalStore():void {
            var userName:String = "Joseph.Labrecque@du.edu";
            var passWord:String = "Adobe_AIR_Rocks!!!";

            var userBytes:ByteArray = new ByteArray();
            userBytes.writeUTFBytes(userName);

            EncryptedLocalStore.setItem("username", userBytes);

            var passBytes:ByteArray = new ByteArray();
            passBytes.writeUTFBytes(passWord);

            EncryptedLocalStore.setItem("password", passBytes);

            traceField.appendText("Data written to EncryptedLocalStore!\n");
            traceField.appendText("\tusername: " + userName + "\n");
            traceField.appendText("\tpassword: " + passWord + "\n\n");
            readEncryptedLocalStore();
        }

        protected function readEncryptedLocalStore():void {
            var userData:ByteArray = EncryptedLocalStore.getItem("username");
            var passData:ByteArray = EncryptedLocalStore.getItem("password");

            traceField.appendText("Data retrieved from EncryptedLocalStore!\n");
            traceField.appendText("\tuserData: " +
userData.readUTFBytes(userData.length) + "\n");
            traceField.appendText("\tpassData: " +
passData.readUTFBytes(passData.length) + "\n\n");

            clearEncryptedLocalStore();
        }

        protected function clearEncryptedLocalStore():void {

            EncryptedLocalStore.reset();

            traceField.appendText("EncryptedLocalStore has been reset.\n");
            traceField.appendText("All sensitive data has been totally cleared!");
        }

    }
}
```

Figure 9-1 demonstrates how the output of these operations appears on a device.

www.it-ebooks.info

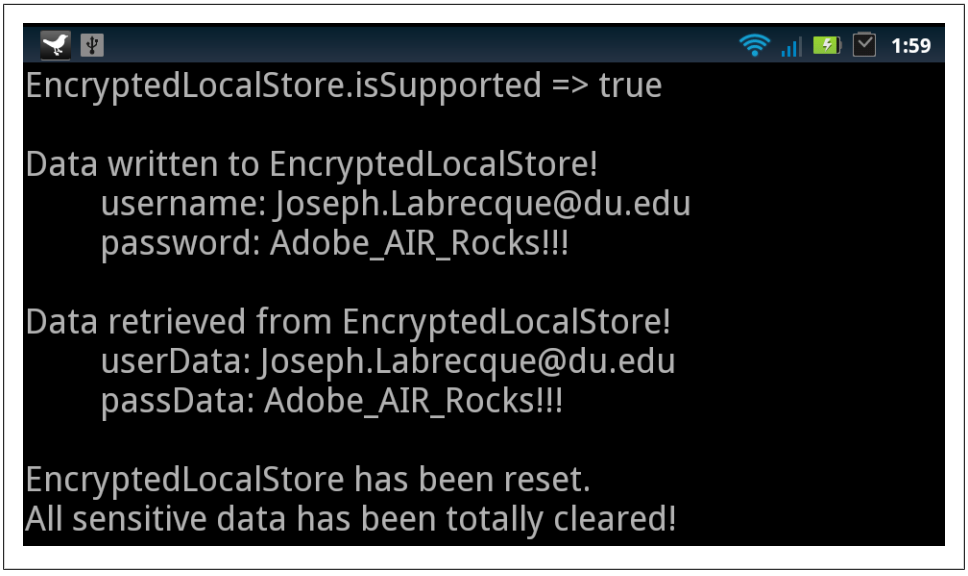


Figure 9-1. Encrypted local store

Protected HTTP Dynamic Streaming and Flash Access Content Protection Support for Mobile

HTTP Dynamic Streaming (*HDS*) was introduced in Flash Media Server 4 and allows the streaming of live or on-demand media over Hyper Text Transfer Protocol (*HTTP*) instead of the Real Time Message Protocol (*RTMP*) family of streaming protocols. This can be very useful if the port used by *RTMP* (usually 1935) is blocked by a network. This technology has now been extended to mobile versions of AIR, as it was previously only available on desktop versions of the runtime.

It is important to note that delivering video over RTMP is still the most secure, robust method of streaming available. HLS simply provides content providers with additional options.

Adobe Flash Access is a Digital Rights Management (*DRM*) system which can be used when deploying content to AIR using Flash Media Server (*FMS*) over *RTMP* or *HTTP*. While previous versions of AIR have supported Flash Access with the desktop runtime only, AIR 3 provides the same level of security on mobile devices.

www.it-ebooks.info

For information on Adobe Flash Access 2.0, please refer to the following URL: *http://www.adobe.com/products/flashaccess/*

When developing an application targeting AIR on mobile devices, the code used is exactly the same as that which has been used to target previous versions of AIR on the desktop. Any applications which are currently set up to handle *DRM* through Flash Access 2.0 should now function exactly the same if encountered with a mobile device which supports AIR 3.

If using the Open Source Media Framework (*OSMF*) with Flash Access, there are a number of events and properties that have been put into place in order to take advantage of this. For more information, please refer to the content at *http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/org/osmf/events/DRMEvent.html*

Secure Random Number Generator

Using the new `flash.crypto.generateRandomBytes()` method, ActionScript developers can generate a set of highly secure, random bytes for use in applications which require cryptographic keys for use in banking, finance, or even in the creation of general-use secure session ids applicable in just about any application which requires a heightened level of security. The actual functions which generate these cryptographically secure bytes actually are generated by the underlying operating system itself and not AIR.

In the example below, we will use this new method to generate a `ByteArray` object containing exactly 1024 randomly generated, cryptographically secure bytes.

```
package {
    import flash.crypto.generateRandomBytes;
    import flash.display.Sprite;
    import flash.text.TextField;
    import flash.text.TextFormat;
    import flash.utils.ByteArray;

    [SWF(width="600", height="500", backgroundColor="#CCCCCC")]

    public class RandomBytes extends Sprite {

        private var traceField:TextField;
        private var randBytes:ByteArray;

        public function RandomBytes() {
            generateDisplayObjects();
            performOperations();
        }
```


www.it-ebooks.info

```
        protected function generateDisplayObjects():void {
            var defaultFormat:TextFormat = new TextFormat();
            defaultFormat.font = "Arial";
            defaultFormat.size = 22;
            defaultFormat.color = 0xFFFFFF;

            traceField = new TextField();
            traceField.backgroundColor = 0x000000;
            traceField.alpha = 0.7;
            traceField.width = stage.stageWidth;
            traceField.height = stage.stageHeight;
            traceField.wordWrap = true;
            traceField.multiline = true;
            traceField.background = true;
            traceField.defaultTextFormat = defaultFormat;
            addChild(traceField);
        }

        protected function performOperations():void {
            traceField.text = "Here come the securely generated random bytes!\n\n";
            randBytes = generateRandomBytes(1024);
            traceField.appendText(randBytes.toString());
        }

    }
}
```

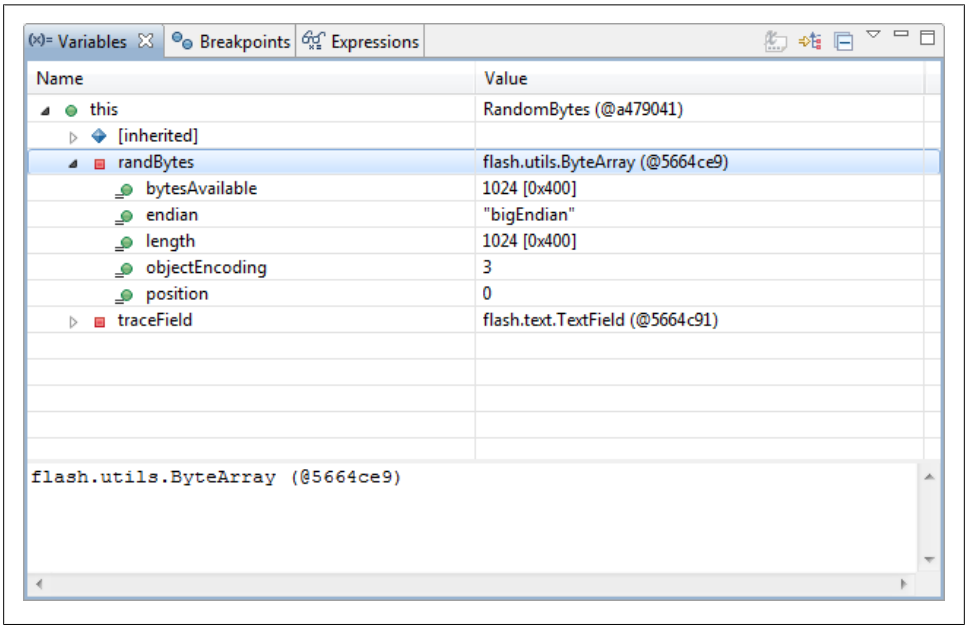


Figure 9-2. Random bytes via Flash Builder debugger

www.it-ebooks.info

By placing a breakpoint within our code, we can view the generated `ByteArray` details within our debugger. In Figure 9-2, we see that the `ByteArray` contains exactly 1024 bytes. This is precisely the number of bytes requested when invoking `flash.crypto.generateRandomBytes(1024)` in our example code.

We are able to request any number of bytes between 1 and 1024 through this new method.

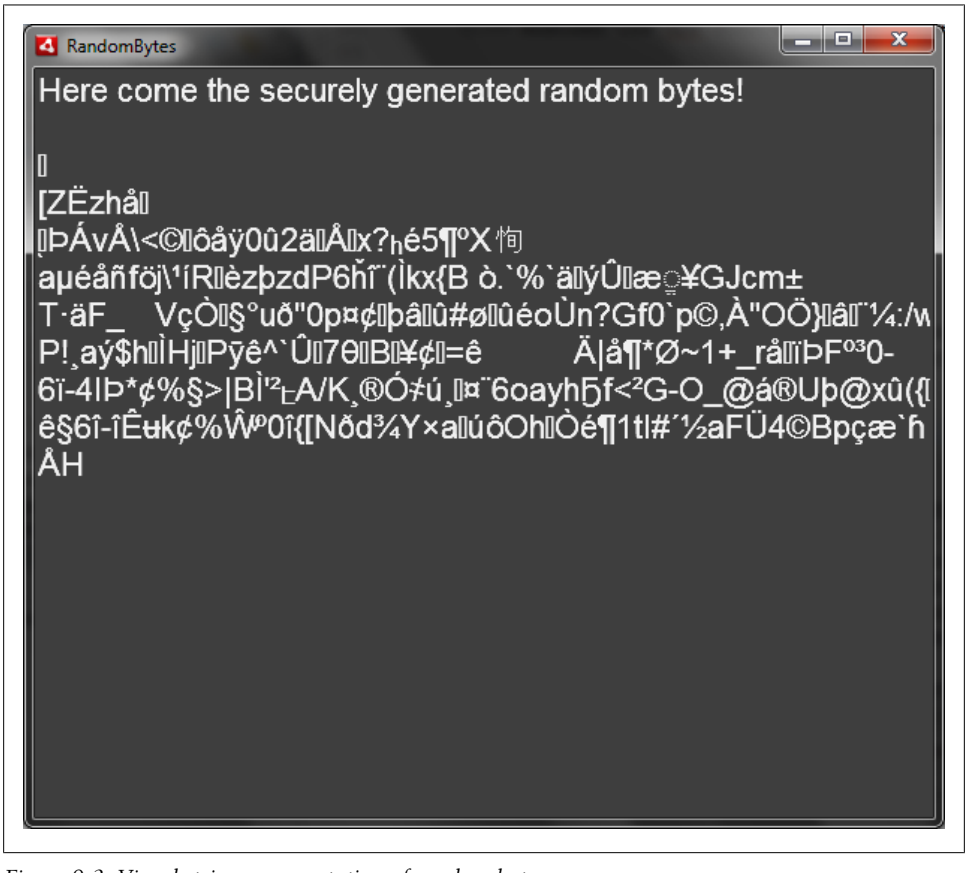


Figure 9-3. Visual string representation of random bytes

www.it-ebooks.info

www.it-ebooks.info

APPENDIX

Additional Resources

We hope you have found this book useful in getting a jump start on understanding and using the new features available in Adobe AIR 3. If you wish to explore further, we recommend the following resources.

What's New in Flash Player 11

AIR 3 has a companion runtime for browser-based applications on desktop and mobile called Flash Player 11. To learn about the new features in Flash Player 11, pick up a copy of *What's New in Flash Player 11* (O'Reilly).

Using Stage3D Frameworks

Adobe has worked closely with a number of frameworks and gaming engines to make sure that `Stage3D` is well supported by a number of projects throughout industry and the community. The following is a sampling.

3D Frameworks

- Alternativa Platform
  http://alternativaplatform.com/en/
- Away3D
  http://www.away3d.com/
- Coppercube
  http://www.ambiera.com/coppercube/
- Flare3D
  http://www.flare3d.com/
- Minko

www.it-ebooks.info

http://aerys.in/minko

- Sophie 3D
  http://www.sophie3d.com/website/index_en.php
- Yogurt3D
  http://www.yogurt3d.com/
- Zest3D
  http://zest3d.digital-glue.com/

2D Frameworks

- Starling
  http://www.starling-framework.org/
- M2D
  https://github.com/egreenfield/M2D
- ND2D
  https://github.com/nulldesign/nd2d

Articles and Resources

Here are some additional resources which are available on the Web.

- Mobile and Devices Developer Center
  http://www.adobe.com/devnet/devices.html
- Flash Platform Game Developer Center
  http://www.adobe.com/devnet/games.html
- Rich Internet application development
  http://www.adobe.com/devnet/ria.html
- Video Technology Center
  http://www.adobe.com/devnet/video.html
- Adobe Evangelists Super Blog
  http://adobeevangelists.com/superblog/
- How to Overlay the AIR SDK for Use With the Flex SDK
  http://kb2.adobe.com/cps/495/cpsid_49532.html
- Native extensions for Adobe AIR
  http://www.adobe.com/devnet/air/native-extensions-for-air.html
- Setting Up Flash Builder 4.5 for Flash 11 and AIR 3 Apps
  http://www.fmsguru.com/showtutorial.cfm?tutorialID=59

www.it-ebooks.info

- Overlay AIR SDK | Flash Professional CS5.5
  http://kb2.adobe.com/cps/908/cpsid_90810.html

www.it-ebooks.info

www.it-ebooks.info

About the Author

Joseph Labrecque is primarily employed by the University of Denver as senior interactive software engineer specializing in the Adobe Flash Platform, where he produces innovative academic toolsets for both traditional desktop environments and emerging mobile spaces. Alongside this principal role, he often serves as adjunct faculty, communicating upon a variety of Flash Platform solutions and general web design and development subjects.

In addition to his accomplishments in higher education, Joseph is the proprietor of Fractured Vision Media, LLC, a digital media production company, technical consultancy, and distribution vehicle for his creative works. He is founder and sole abiding member of the dark ambient recording project "An Early Morning Letter, Displaced" whose releases have received international award nominations and underground acclaim.

Joseph has contributed to a number of respected community publications as an article writer and video tutorialist and is author of the *Flash Development for Android Cookbook* (2011 Packt Publishing - ISBN: 1849691428), *What's New in Adobe AIR 3* (2012 O'Reilly - ISBN: 9781449311087), *What's New in Flash Player 11* (2012 O'Reilly - ISBN: 9781449311100), and co-author of *Mobile Development with Flash Professional CS5.5 and Flash Builder 4.5: Learn by Video* (2011 Adobe Press - ISBN: 0321788109).

He regularly speaks at user group meetings and industry conferences such as Adobe MAX, FITC, D2W, and a variety of other educational and technical conferences. In 2010, he received an Adobe Impact Award in recognition of his outstanding contribution to the education community. He has served as an Adobe Education Leader since 2008 and is also a 2011 Adobe Community Professional.

www.it-ebooks.info

www.it-ebooks.info